1985-12

Micro-computer Modeling and Simulation of Conformal Acoustic Array Performance Array Element Degradation Effects

Fleurant, Sylvain J.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MICRO-COMPUTER MODELING AND SIMULATION OF CONFORMAL ACOUSTIC ARRAY PERFORMANCE AND ARRAY ELEMENT DEGRADATION EFFECTS

by

Sylvain J. Fleurant

December 1985

Thesis Co-advisor: O. B. Wilson
Thesis Co-advisor: R. W. Hamming
Thesis Co-advisor: L. J. Ziomek

~~Distribution limited to U.S. Government Agencies and their Contractors; Critical Technology; December 1985. Other requests for this document must be referred to the Superintendent, Naval Postgraduate School, Code 043, Monterey, California 93943-5004 via the Defense Technical Information Center, Cameron Station, Alexandria, Virginia 22304-6145.~~

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

[redacted]

[redacted]

December 6, 2017

SUBJECT: Change in distribution statement for *Micro-Computer Modeling and Simulation of Conformal Acoustic Array Performance Array Element Degradation Effects* – December 1985.

1. Reference: Fleurant, Sylvain J. *Micro-Computer Modeling and Simulation of Conformal Acoustic Array Performance Array Element Degradation Effects*. Monterey, CA: Naval Postgraduate School, Engineering Acoustics Academic Committee; Department of Computer Science, December 1985.

2. Upon consultation with NPS faculty, the School has determined that this thesis may distributed externally with no limitations effective November 29, 2017.

University Librarian
Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS DOWNGRADED APPROVED FOR PUBLIC RELEASE
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT ~~Distribution limited to U.S. Government Agencies and their Contractors; Critical Technology; December 1985. Other requests~~
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b. OFFICE SYMBOL (*If applicable*): Code 61	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*): Monterey, California 93943 - 5100	7b. ADDRESS (*City, State, and ZIP Code*): Monterey, California 93943 - 5100
8a. NAME OF FUNDING / SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
MICRO-COMPUTER MODELING AND SIMULATION OF CONFORMAL ACOUSTIC ARRAY PERFORMANCE AND ARRAY ELEMENT DEGRADATION EFFECTS

12. PERSONAL AUTHOR(S)
Fleurant Sylvain J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Master's Thesis	FROM _____ TO _____	1985, December	290

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Aperture function, beam pattern, Conformal acoustic array, directivity function, Hydrophone, relative sensitivity, software simulation.

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

A far-field beam pattern or directivity function of a conformal acoustic array can be generated by performing a three dimensional spatial discrete Fourier transform of the array's aperture function. The beam pattern is a function of frequency, and of both spherical angles of direction. It is strongly dependent on the aperture window, on the applied time delays, and the geometrical configuration of the array. A user-friendly micro-computer software program was created to study the actual array design and the degraded array. The program generates tabular and graphical data in a short period of time (3 min.) for a given frequency, bearing angle (MRA), speed of sound, and configuration. Finally, this thesis shows that the degradation of even a few elements seriously influences the sensitivity and the shape of the beam pattern, and the user must draw conclusions on an individual basis.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL: O.B. Wilson	22b. TELEPHONE (*Include Area Code*): (408) - 646 - 2729	22c. OFFICE SYMBOL: 61Wl

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

3. ~~(ctd) . . . for this document must be referred to Superintendent, Code 043, Naval Postgraduate School, Monterey CA 93943-5100 via the Defense Technical Information Center, Cameron Station, Alexandria, Virginia, 22304-6145.~~

Micro-Computer Modeling and Simulation of Conformal Acoustic Array Performance and Array Element Degradation Effects

by

Sylvain J. Fleurant
Captain, Canadian Armed Forces
B. Sc. (Physics), Universite de Montreal, 1979

Submitted in partial fulfillment of the
requirements of the degrees of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS
and
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1985

Author: [redacted]
Sylvain J. Fleurant

Approved by: [redacted]
O. B. Wilson, Thesis Co-Advisor

[redacted]
R. W. Hamming, Thesis Co-Advisor

[redacted]
L. E. Ziomek, Thesis Co-Advisor

[redacted]
James V. Sanders, Chairman
Engineering Acoustics Academic Committee

[redacted]
V. Y. Lum, Chairman
Department of Computer Science

[redacted]
John Dyer, Dean of Science and Engineering

[redacted]
Kneale T. Marshall
Dean of Information and Policy Sciences

ABSTRACT

A far-field beam pattern or directivity function of a conformal acoustic array can be generated by performing a three dimensional spatial discrete Fourier transform of the array's aperture function. The beam pattern is a function of frequency, and of both spherical angles of direction. It is strongly dependent on the aperture window, on the applied time delays, and the geometrical configuration of the array. A user-friendly micro-computer software program was created to study the actual array design and the degraded array. The program generates tabular and graphical data in a short period of time (3 min.) for a given frequency, bearing angle (MRA), speed of sound, and configuration. Finally, this thesis shows that the degradation of even a few elements seriously influences the sensitivity and the shape of the beam pattern, and the user must draw conclusions on an individual basis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to acknowledge the interest, time, and effort provided by my thesis co-advisors, Professor O.B. Wilson, Professor H.W. Hamming, and Dr. L.J. Ziomek. Dr. Wilson's insight and quidance were appreciated throughout the development of this project. Professor H.W. Hamming's support was invaluable. His broad knowledge and experience of both engineering and computer science were available at critical times. He contributed substantially to the successful completion of this thesis through his encouragement and insightful reviews of early drafts. Dr. L.J. Ziomek contributed greatly to the successful development of the theory used and is responsible to a great degree for the fact that this project evolved into such a rewarding endeavour.

The support of the Naval Sea Systems Command Code 05NE58(C. Allens) for this project is acknowledged. I gratefully acknowledged the contribution of Mr. Dudley Baker of the Applied Research Laboratories at the University of Texas, Austin, for the important data supplied at a critical time during software development. Furthermore, I especially thank Ms Parma Yarkin, who is truly an "artist of the pencil". Parma's multiple revisions greatly improved the quality of this thesis.

Most importantly, I thank my wife, Annick, and my sons, Guillaume and Sebastien, without whose constant support and understanding this thesis might never have been completed.

I. INTRODUCTION

A. BACKGROUND

A ship-mounted SONAR system consists of an acoustic array and the accompanying electronic equipment. It is used to form beams and provides signal processing of the received signals to permit localization, speed evaluation, and, eventually, classification of sound sources in the ocean.

An acoustic array is formed by an ensemble of hydrophones or projectors which are generally electroacoustic transducers. They transform or transduce acoustic energy into electric energy when used as receivers, or vice-versa if used as projectors. Such an array can be organized geometrically in order to provide directional projection or reception of sound in both the vertical (polar) and the horizontal (azimuthal) direction.

One of the most important parts in the actual SONAR is the hydrophone. Its receiving (and transmitting) properties may change with time and operational conditions. This is especially true when hydrophones are mounted in a ship's SONAR array where they are exposed to the rough ocean environment. Array theory predicts that a change in the properties of one or more transducers usually translates into a degradation of array performance. However, the amount of performance degradation is not easily predicted by using a back-of-the-envelope calculation.

These SONAR systems usually perform very well within the limitations of their designs so long as no changes occur in the different components; i.e. decalibration, aging, loss of power, high internal noise, element degradation, elements breaking, etc.

For the past three or four decades, military and civilian personnel using passive hull mounted SONAR arrays

mounted on submarines or ships have had to rely on a typical design frequency response or directivity function provided for an "undegraded" or "perfect" array in order to estimate the bearing of any underwater object being listened to.

A problem arises if some elements of the array break, degrade in performance, or are simply not performing properly; how can one determine the effects on the performance of the array and what is the resulting directivity function?

B. OBJECTIVES

The purpose of this thesis is to address the problem of the degraded beam pattern or directivity function. Knowing the actual up-to-date complex sensitivity (or complex weights), i.e., both normalized amplitude and phase angle of the sensitivity of each hydrophone forming the conformal array (hull-mounted acoustic array), what is the resulting degraded response of the array as a function of frequency and angles of direction?

The following objectives are proposed:

1. Develop (or derive) the appropriate exact algorithms for computing the far-field beam pattern of the conformal array using arrray theory;
2. Design and develop a stand-alone user-friendly software program, to be run on a micro-computer with graphics capability, which will simulate the performance of a given fixed-element conformal array mounted on the hull of a submarine using the actual complex sensitivity of the elements;
3. Produce both tabular and graphical representations of the normalized theoretical, i.e., design, and degraded far-field beam patterns showing the main and secondary lobes, relative amplitude, changes in sensitivity (detection range), steering changes, and beam deformation;

4. Investigate some typical cases of array performance degradation due to typical hydrophone failures such as cracking, flooding, and removal, along with padding of degraded hydrophones (using added capacitors); and
5. A Users Manual will be produced. The design requirements, design specifications, and relevant information used by the maintainer will be part of this thesis.

It is important to mention that this thesis is part of a larger project about performance monitoring of the DT-276 Hydrophone and BQR-7 array presently used by the US Navy. Several students at the US Naval Postgraduate School have developed an automated test and evaluation system using micro-computers as instrument controllers along with other measuring devices. Their results along with those from other teams will provide part of the data required for this thesis. More details will be provided in the following chapters.

C. DEFINITIONS

In order to help make clear the results of this thesis, some definitions are now given. An acoustic array is a directional receiver (or projector). It may reduce the effects of noise or interfering signals coming in some direction while enabling the estimation of the bearing of a sound source coming from another direction. [Ref. 1: Chap. 9]

Therefore, it is important to know the functional dependence of the array's sensitivity on azimuth angle ψ, elevation angle θ, frequency f, and speed of sound c. Generally, acousticians use relative amplitude of pressure as the unit of measurement of a signal. Here the general definition of a far-field beam pattern is the angular functional dependence of the normalized sound pressure amplitude

at a large, fixed distance from the source, where "large distance" will be explained in the next chapter [Ref. 1].

Other names for the far-field beam pattern are pattern function, directivity function, directivity pattern, and even radiation pattern (for an emitter). Typical features are its extrema (i.e., minima and maxima) angles, beamwidth, directivity (& directivity index), grating lobes, etc.

Knowing the expression of the directivity function , it is possible to evaluate the acoustic pressure amplitude in the far field which is written as

$$P(r,\theta,\psi) = Pax(r)\, D(\theta,\psi) \qquad (1.1)$$

where $$D(\theta,\psi) = D(f,\theta,\psi)$$

is the directivity function, and Pax(r) = on-axis pressure, i.e., far-field pressure on the main acoustic axis. Therefore, values of the acoustic pressure at other angles can be computed using the directivity function and the axial pressure.

D. FORMAT OF THE THESIS

In Chapter 2, the basic theory of beam patterns is developed, along with relevant assumptions and major features. Aspects such as time delay, sensitivity, physical location of the elements, symmetry, and diffraction are addressed.

Chapter 3 provides the details of the first three phases of software development: analysis (planning), design, and implementation of the software program. It also provides details about the approach used, the prototype developed, the software requirements, the external and the detailed design, along with the coding style used and documentation procedures.

Chapter 4 is about the last phase of software development, which is the system testing. Only the verification testing will be done. No validation testing will be done since no data are available for comparison. Additionally, the last phase of the life-cycle of the software program, software maintenance, will not be covered.

Finally, chapter 5 will provide an analysis of some of the results obtained with standard data due to degradation, and what to expect if an element is absent or partially degraded (changes in phase and in amplitude of its sensitivity). The results are interpreted in context. A section will list recommendations for future improvements, along with a conclusion.

Finally, throughout the text, the words "far-field beam pattern" and "directivity function" are used synonymously.

II. THEORY

A. DESCRIPTION OF THE TASK

An ensemble of hydrophones mounted on the hull of a ship forms an acoustic array which is called a conformal array , as opposed to cylindrical or spherical arrays. Usually, only a portion of the total array is used when the main lobe is steered in a specific direction in order to minimize the problem of accounting for diffraction around the hull for hydrophones (also called elements) not directly exposed to the incoming signal. This is the case for the BQR-7 array and for many similar systems.

For a fixed conformal "sub-array" of the total array (shaped like a submarine or surface vessel), i.e., at a given bearing selection, the objective is to define the far-field beam pattern as a function of the angles of direction (horizontal and vertical), the speed of sound, the frequency, the mode of operation (i.e., aperture window (shading) or the mode SUM and DIFF), the implemented time delay, and the complex sensitivity of every element.

As in any theoretical development, numerous assumptions will be made. Here assumptions described by Dr L. J. Ziomek [Ref. 2: Chap 1]. will be used. It is assumed that the speed of sound in the medium (i.e., the ocean) is constant over the entire span of the array, which is realistic since the maximum length of the BQR-7 array is usually about 14 meters.

The linear, inhomogeneous, scalar wave equation solution is used, which excludes nonlinear effects.

Internal wave motion will not be considered. No mutual coupling is considered since no data are available. That is, each hydrophone in the array will be regarded as acting independently of all others [Ref. 3: Chap 5]. The elements

are considered to be point receivers, i.e., lumped-element acoustics is used. This greatly simplifies the defining of the far-field beam pattern. Indeed, in most cases the wavelength is much larger than the dimensions of an element. For example, at 1000 Hz and at a speed of 1500 m/sec, the wavelength is 150 cm, which is much larger than the largest portion of an element (about 15 cm). Therefore, it is a valid assumption.

The most critical assumption is the exclusion of any diffraction of incoming plane waves around the hull of the submarine which changes the value of the measured signal at a given point. Diffraction is a function of frequency, incoming angle of arrival, submarine shape, etc. This important assumption will be addressed later but must be made due to the complexity of the shape, the large diversity of devices present in the submarine, and the lack of measured data. Chapter 3 will discuss the features allowing future implementation of the diffraction of the signal around the submarine.

B. SOME DEFINITIONS AND BASIC CONCEPTS

1. Far-Field or Fraunhofer Zone

The far-field or the Fraunhofer Zone is the region where the beam pattern is independent of the range between the source and the receiver. There are different ways to define the beginning of that region. According to Ziomek [Ref. 2: p. 38], an observation point at the range r will be considered to be in the far-field if

$$r > \frac{\pi R^2}{\lambda} \qquad (2.1)$$

where R is the maximum radial dimension of the array. For a smaller range, the zone becomes the near-field or Fresnel zone. It should be mentioned that in the near-field, the beam pattern is a function of the range r.

Steinberg [Ref. 3: p. 12] uses a slightly different definition which is defined by equation 2.2

$$r > \frac{R^2}{\lambda} \qquad (2.2)$$

It is a smaller distance.

Kinsler and Frey [Ref. 4: p. 188] use a similar one but even shorter;

$$r > \frac{R^2}{4\lambda} \qquad (2.3)$$

Therefore the most conservative range to the far-field is given by equation 2.1 . For the BQR-7E array, R is approximaly 7.5 meters (25 feet). For a frequency f of 1000 Hz and speed of sound c of 1500 m/s, the wavelength λ given by

$$\lambda = \frac{c}{f} = 1.5m$$

yields a distance to the far-field of about 118 meters (387 feet) as calculated from equation 2.1 . At a lower frequency, say 100 Hz, the far-field begins at 11.8 meters (38.7 feet).

Usually targets or sources dealt with by a hull-mounted array are more distant than 120 meters.

2. Aperture Function

As for any physical system, the conformal array possesses a complex frequency response. It is commonly referred to as the receiver complex aperture function or simply the aperture function [Ref. 2,3: Chap 4 and Chap 5] It is a function of frequency and the spatial location r of the elements relative to the center of the array. The symbol to be used for the aperture function is A(f, x, y, z).

In the far-field, the aperture function and the far-field beam pattern form a spatial Fourier transform pair. To know the aperture function is to know the beam pattern.

3. Coordinate System

The coordinate system to be used here will be the convention of Ziomek [Refs. 2,3: p. 33 and p. 15]. The origin is taken to be the center of the total array. The array lies in the xy plane. The normal to the array is the z axis. Figure 2.1 shows the coordinate systems to be used throughout this thesis.

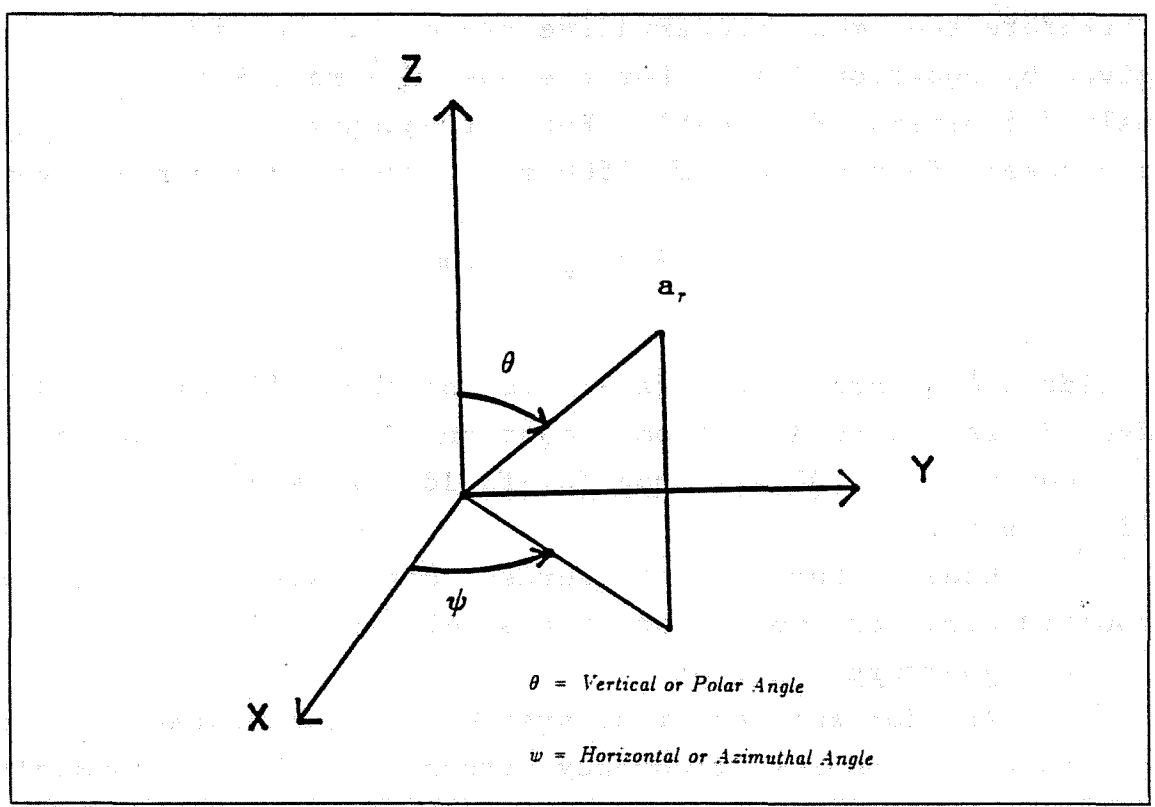

Figure 2.1 Coordinate System.

Figure 2.1 shows a vector, say $\mathbf{a}_r$, which is given by

$$\mathbf{a}_r = u\mathbf{x} + v\mathbf{y} + w\mathbf{z} \tag{2.4}$$

where

$$u = \sin\theta \cos\psi \tag{2.5}$$

$$v = \sin\theta \sin\psi \tag{2.6}$$

and

$$w = \cos\theta \tag{2.7}$$

are the direction cosines with respect to the positive X, Y, and Z axis, respectively. Note that

$$u^2 + v^2 + w^2 = 1 \tag{2.8}$$

Any point can then be localized by a vector $\mathbf{r}$, such that

$$\mathbf{r} = x\mathbf{x} + y\mathbf{y} + z\mathbf{z} \tag{2.9}$$

$$= r\mathbf{a}_r \tag{2.10}$$

4. Spatial Frequencies

The spatial frequencies are three quantities defined by [Ref. 2: p. 36]:

$$f_x = \frac{u}{\lambda} \tag{2.11}$$

$$f_y = \frac{v}{\lambda} \tag{2.12}$$

and

$$f_z = \frac{w}{\lambda} \tag{2.13}$$

and u, v, and w are as in equations 2.5, 2.6, and 2.7 . Note that these spatial frequencies are in cycles per meter. They are directly related to the wave number components by the following expressions:

$$k_x = 2\pi f_x \tag{2.14}$$

$$k_y = 2\pi f_y \tag{2.15}$$

and

$$k_z = 2\pi f_z \tag{2.16}$$

Additionally, the components of the wave number vector $\mathbf{k}$, along the coordinate axes must satisfy the relation:

$$k^2 = \mathbf{k}\mathbf{k}^* = k_x^2 + k_y^2 + k_z^2 \tag{2.17}$$

where $\mathbf{k}^*$ = complex conjugate of $\mathbf{k}$.

5. Complex Weights

The complex weights are used for amplitude and phase weighting, as discussed later. The complex weights allow the characterization of individual "identical" transducers by showing their contribution or "excitation" in the array in a normalized way . For example, an amplitude of 0 dB implies full weighting of the element to the array while -3

dB corresponds to one-half of its capability (this excludes the effects of phasing, shading, steering, etc.).

In our application, the relative complex sensitivities will be used as complex weights. They must, of course, be normalized to a standard such as the complex weights of a "perfect" hydrophone, which should have an amplitude of 0 dB (or magnitude of 1.0) and a phase of 0 degrees.

Moreover, the complex weights can be filtered through an amplitude window to create the proper phased array beam pattern. The theory will separate them clearly. The term "complex weights" will refer to the normalized complex sensitivities throughout the text.

C. THE GENERAL APPROACH FOR FINDING THE BEAM PATTERN

The far-field beam pattern is simply related to the aperture function through the Fourier Transform. The far-field directivity function can then be written as [Refs. 2,3: p.38 and Chap. 5]:

$$D(f,f_x,f_y,f_z) \equiv \mathbf{F}_x\ \mathbf{F}_y\ \mathbf{F}_z\ (A(f,x,y,z)) \qquad (2.18)$$

where the triple product of F operators represents a three-dimensional spatial Fourier transform. Equation 2.18 can be rewritten as:

$$A(f,x,y,z) = \mathbf{F}_{f_x}^{-1}\,\mathbf{F}_{f_y}^{-1}\,\mathbf{F}_{f_z}^{-1}\,(D(f,f_x,f_y,f_z)) \qquad (2.19)$$

Since the spatial frequencies are functions of the horizontal and vertical angles, equation 2.18 can be rewritten as

$$D(f,f_x,f_y,f_z) = D(f,\theta,\psi) \qquad (2.20)$$

The following equation is the complete expression for equation 2.18

$$D(f,\theta,\psi) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A(f,x,y,z)\, e^{+j2\pi(f_x x + f_y y + f_z z)}\, dx\, dy\, dz \quad (2.21)$$

The next step is to develop an expression for the aperture function.

It is important to mention the existence of some useful properties of the Fourier transform as derived in Oppenheim and Schafer [Ref. 5: Chap 3], and also by Brigham [Ref. 6: Chap 2].

Linear addition is conserved in both space and the spatial frequency domain. A space shift corresponds to a phase (spatial frequency) shift. An even function becomes a real function after a direct Fourier transform, while an odd function produces a pure imaginary function.

Multiplication in the space domain corresponds to a convolution in the spatial frequency domain. The converse is also true.

The previous properties apply to both integral and discrete Fourier transforms.

D. ARRAY, A SAMPLED APERTURE FUNCTION

An array is an aperture excited (or being excited) only at points or in localized areas. It consists of discrete sources or receivers (hydrophones) called elements.

Most books treating this theory deal with convenient symmetric arrays such as line arrays, planar (flat) arrays, spherical, or cylindrical arrays with the elements equally spaced along each axis (the spacing can be different between 2 axes). A great deal of simplification can then be made.

Moreover, the resulting far-field beam pattern is simply a three-dimensional discrete spatial Fourier transform (DFT) (since we sample equation 2.21). Usually, odd numbered element arrays are used to simplify the resulting equation still further. One can then use a fast Fourier transform (FFT) computer algorithm to compute the discrete Fourier

transfom. This is adequate for the previously mentioned configuration but can result in imprecision or long computer time (due to the large sampling rate required by the FFT) for an unequally spaced array.

Furthermore, a suitable closed form expression can be derived for a standard shape like a dipole, a line array, or a rectangular flat array but it cannot be made for an arbitrary conformal array. The accuracy of the result is important for a conformal array. Therefore, exact DFTs will be done to minimize CPU time for unequally spaced array.

To explain the procedure for developing a beam pattern, a simple example using symmetry is given. Consider a line array composed of an odd number N of identical, complex-weighted, equally spaced elements centered at the origin, lying along the X-axis (with one element at the origin). The complex frequency response or complex aperture function A(f, x, y, z) of this array is simply expressed by the following equation [Ref. 2: p. 96]:

$$A(f,x,y,z) = A(f,x) = \sum_{n=\frac{-(N-1)}{2}}^{\frac{(N-1)}{2}} c_n \, e(f,x-nd) \qquad (2.22)$$

where n represents one element, and N is the total number of elements, d is the interelement spacing in meters, c_n is the complex weight and e(f, x) is the complex frequency response of the identical elements, also known as the element function.

For a point source, we have

$$e(f,x) = \delta(x) \qquad (2.23)$$

So A(f, x) can be written as

$$A(f,x) = s(x) * e(f,x) \qquad (2.24)$$

where

$$s(x) = \sum_{n=\frac{-(N-1)}{2}}^{\frac{(N-1)}{2}} c_n \, \delta(x-nd) \tag{2.25}$$

and the asterix (*) indicates a convolution with respect to x.

The function s(x) is basically the complex frequency response or aperture function of an equivalent linear array of N odd complex-weighted point sources.

Using equation 2.18, the far-field beam pattern of equation 2.24 becomes

$$D(f,f_x,f_y,f_z) = D(f,f_z) \tag{2.26}$$

$$= S(f_z)\,E(f,f_z) \tag{2.27}$$

where

$$E(f,f_z) = \mathbf{F}_x(e(f,x)) \tag{2.28}$$

and

$$S(f_z) = \mathbf{F}_x(s(x)) \tag{2.29}$$

Equation 2.27 is called the Product Theorem for a linear array. It states that the far-field beam pattern of a line array of identical elements (complex-weighted) is equal to the product of the far-field beam pattern E of one of the elements and the far-field beam pattern S of an equivalent linear array of complex-weighted point sources.

If the linear array is composed of point sources, that is, if

$$e(f,x) = \delta(x) \tag{2.30}$$

then

$$E(f,f_x) = 1 \tag{2.31}$$

since

$$E(f,f_x) = \mathbf{F}_x\{e(f,x)\} \tag{2.32}$$

$$= \int_{-\infty}^{\infty} e(f,x)\exp(+j2\pi f_x x)dx \tag{2.33}$$

As a result, equation 2.27 becomes the following equation for point sources

$$D(f,f_x) = S(f_x) \tag{2.34}$$

Therefore, using equation 2.31 , equation 2.27 becomes the following equation for an odd-numbered line array of point sources

$$D(f,f_x) = S(f_x)$$

$$= \sum_{n=-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} c_n \exp(+j2\pi f_x nd) \tag{2.35}$$

This expression is easy to compute.

In general, the complex weights can be expressed as

$$c_n = a_n \exp(+j\theta_n) \tag{2.36}$$

With the preceding equation, equation 2.35 becomes

$$D(f, f_x) = \sum_{n=-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} a_n \exp[+j(2\pi f_x nd + \theta_n)] \tag{2.37}$$

It is then possible to implement an amplitude weight by applying an amplitude window to the array. This process is called amplitude shading or amplitude-weighting.

Additionally, the phase terms of every element in equation 2.37 can be modified so that the array scans a region. This is done by varying the relative phase between array elements. In order to obtain an "undeformed" beam pattern steered to a specific listening angle (vertical or horizontal), a linear time delay must be used, where the undeformed beam pattern has the same shape as one without the additional relative phase. The following terms can be added for each n:

$$\theta_n = 2\pi f \tau_n \tag{2.38}$$

with

$$\tau_n = -u' \frac{nd}{c} \tag{2.39}$$

where u' is the direction cosine, i.e., the listening vertical and horizontal angles at which the array is steered.

Equation 2.38 corresponds to the following phases:

$$\theta_n = -2\pi f_x' nd \tag{2.40}$$

where

$$f_x' = \frac{\sin\theta' \cos\psi'}{\lambda} \tag{2.41}$$

Using equation 2.37 with the phase term $\theta_n = 0$, we obtain a new expression that can be called the unsteered or old directivity function:

$$S(f_x) = \sum_{n=-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} a_n \exp[+j(2\pi f_x nd)] \tag{2.42}$$

Again using equation 2.37 with the phase term of equation 2.40 we obtain another expression:

$$S'(f_x) = \sum_{n=-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} a_n \exp[+j(2\pi(f_x - f_x')nd)] \tag{2.43}$$

or

$$S'(f_x) = S(f_x - f_x') \tag{2.44}$$

This new beam pattern can be expressed as the old one but with a difference in phase (or in angle). For the line array, using equation 2.42, the maximum occurs at

$$f_x = 0$$

It is assumed that no additional main lobes (grating lobes) are present. However, with the other equation, i.e. 2.43, the maximum occurs at

$$f_x = f_x'$$

This process is called beam steering where the array becomes a phased-array. The far-field beam pattern can then be steered in the direction:

$$u = u'$$

in direction cosine space when the line array of point sources is phase weighted according to equations 2.38 and 2.39, or with equations 2.40 and 2.41 .

Finally, the computation can be greatly accelerated by remarking that equation 2.35 is in the form of a discrete spatial Fourier transform (DSFT). A fast Fourier transform (FFT) algorithm can then be used to compute it by using the following expression:

$$f_x = mf_{x_s} \quad , \quad m = 0,1,\ldots,M-1 \tag{2.45}$$

where

$$f_{x_s} = \frac{1}{Md} \tag{2.46}$$

and

$$M = \left(\frac{N-1}{2}\right) + 1 \tag{2.47}$$

Thus S is evaluated only at the discrete spatial frequencies defined by equation 2.45 . If we define [Refs. 2,5: p. 113 and p. 89],

$$W_M = \exp\left(+j\frac{2\pi}{M}\right) \tag{2.48}$$

and if

$$c_{-n} = c_n^* \tag{2.49}$$

as is generally the case, then equation 2.35 becomes

$$D(f, f_z) = \sum_{n=0}^{M-1} c_n \, W_M^{mn} \tag{2.50}$$

where m = 0, 1, ... , M - 1 and where

$$c_0 = \frac{a_0}{2} \tag{2.51}$$

Equation 2.50 is standard in most signal processing books, and can easily be implemented in a program [Refs. 5,6: Chap 6 and Chap 10 respectively].

An FFT is efficient since the amount of computation, hence computation time, required by the direct computation of equation 2.37, which is approximately proportional to M squared, is reduced to M log M by using equation 2.50 arranged in a clever fashion. The reason is that most FFT algorithms, such as decimation-in-time and decimation-in-frequency, are based upon the decomposition of the discrete Fourier transform of a sequence of length M into successively smaller discrete Fourier transforms [Ref. 5: p. 286].

Ziomek [Ref. 2: Section 4.1-4] shows the required adjustment to be applied from equation 2.45 to equation 2.50 for a typical FFT algorithm, which requires the total number M to be equal to an integer power of 2, that is

$$M = 2^b \quad , \; b = 1,2,3,... \tag{2.52}$$

This is done by "padding with zero". Indeed, the more padding the more interpolation can be done between the points, i.e., the better the beam pattern curve definition.

The accuracy of the data does not change with padding but only interpolated data are computed.

This very long example shows one approach that can be used to produce the far-field beam pattern of any array as long as some symmetry can be used.

Finally, a missing element can be simulated by replacing its corresponding amplitude weights by zero.

E. THE VOLUME ARRAY - ONE APPROACH

Let us follow a similar approach for the analysis of a volume array (3 dimensional). Ziomek [Ref. 2: Section 4.7] developed expressions for an array of identical complex-weighted point sources lying on the surface of a cylinder, i.e., a cylindrical array, and for an array of identical, complex-weighted point sources lying on the surface of a sphere, i.e., a spherical array.

Consider a volume array (rectangular) made of M x N x P odd number of identical complex-weighted elements. Some definitions can immediately be made;

$$d_x = \textit{constant interelement spacing in the X direction} \tag{2.53}$$

$$d_y = \textit{constant interelement spacing in the Y direction} \tag{2.54}$$

$$d_z = \textit{constant interelement spacing in the Z direction} \tag{2.55}$$

where

$$d_x \neq d_y \neq d_z$$

in general, and

$$m = \textit{Index for the element-coordinate along X} \tag{2.56}$$

$$n = \textit{Index for the element-coordinate along Y} \tag{2.57}$$

p = *Index for the element-coordinate along Z* (2.58)

Using equation 2.18, it is again necessary to find the expression of the aperture function A(f, x, y, z) in order to compute the far-field beam pattern.

An expression similar to equation 2.24 can be deduced, such that

$$A(f,x,y,z) = s(x,y,z) \;***\; e(f,x,y,z) \tag{2.59}$$

where e(f, x, y, z) is the complex frequency response of one of the identical complex-weighted elements (again, it is called the element function). The triple asterix (*) denotes a three-dimensional convolution with respect to x, y, and z.

If and only if the elements are all equally spaced along each axis, an expression for s(x, y, z) can be defined as

$$s(x,y,z) = \sum_{m=-\frac{(M-1)}{2}}^{\frac{(M-1)}{2}} \sum_{n=-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} \sum_{p=-\frac{(P-1)}{2}}^{\frac{(P-1)}{2}} c_{mnp}\, \delta(x-md_x)\, \delta(y-nd_y)\, \delta(z-pd_z) \tag{2.60}$$

Again, s(x, y, z) is the complex frequency response or complex aperture function of an equivalent volume array of complex-weighted point sources.

Therefore, using equations 2.18 and 2.59 the far-field beam pattern can be expressed as

$$D(f,f_x,f_y,f_z) = \mathbf{F}_x\, \mathbf{F}_y\, \mathbf{F}_z\, [s(x,y,z) \;***\; e(f,x,y,z)] \tag{2.61}$$

or

$$D(f, f_x, f_y, f_z) = S(f_x, f_y, f_z)\, E(f, f_x, f_y, f_z) \qquad (2.62)$$

where

$$S(f_x, f_y, f_z) = \mathbf{F}_x\, \mathbf{F}_y\, \mathbf{F}_z\, [s(x, y, z)] \qquad (2.63)$$

and

$$E(f, f_x, f_y, f_z) = \mathbf{F}_x\, \mathbf{F}_y\, \mathbf{F}_z [e(f, x, y, z)] \qquad (2.64)$$

Equation 2.62 is called the product theorem for a volume array.

For a point source element, the expression for the complex frequency response e(f, x, y, z) is

$$\begin{aligned} e(f, x, y, z) &= \delta(x, y, z) \\ &= \delta(x)\, \delta(y)\, \delta(z) \end{aligned} \qquad (2.65)$$

Using equation 2.64, the result is given by :

$$E(f, f_x, f_y, f_z) = 1 \qquad (2.66)$$

So for an array of point source elements, equation 2.62 becomes

$$E(f, f_x, f_y, f_z) = S(f_x, f_y, f_z) \qquad (2.67)$$

Returning to the M x N x P volume array, using equation 2.60, which expresses s(x, y, z), and with equation 2.67, the far-field beam pattern of the array becomes

$$D(f, f_x, f_y, f_z) = \sum_{m=-M'}^{M'} \sum_{n=-N'}^{N'} \sum_{p=-P'}^{P'} c_{mnp} \exp(+j2\pi(f_x m d_x + f_y n d_y + f_z p d_z)) \quad (2.68)$$

where the complex weights are defined, in general, as

$$c_{mnp} = a_{mnp} \exp(+j\theta_{mnp}) \quad (2.69)$$

and where

$$M' = \left(\frac{M-1}{2}\right) \quad (2.70)$$

$$N' = \left(\frac{N-1}{2}\right) \quad (2.71)$$

$$P' = \left(\frac{P-1}{2}\right) \quad (2.72)$$

Basically, the procedure followed is similar to the one used for the line array. Without any derivation, it is possible to steer the array using the proper time delays by adding phase weights, and it is also possible to shade the array by using an amplitude window (3-dimensional). Moreover, the far-field beam pattern expressed by equation 2.68 is a three-dimensional discrete spatial Fourier transform.

It is then possible to use an FFT algorithm to compute the beam pattern by doing the proper sampling in the spatial frequency domain. As for the line array, the following expression can be used:

$$f_x = q f_{x_s} \quad , \quad q = -M', \ldots, 0, \ldots, M' \tag{2.73}$$

$$f_y = r f_{y_s} \quad , \quad r = -N', \ldots, 0, \ldots, N' \tag{2.74}$$

$$f_z = s f_{z_s} \quad , \quad s = -P', \ldots, 0, \ldots, P' \tag{2.75}$$

where

$$f_{x_s} = \frac{1}{M\ d_x} \tag{2.76}$$

$$f_{y_s} = \frac{1}{N\ d_y} \tag{2.77}$$

$$f_{z_s} = \frac{1}{P\ d_z} \tag{2.78}$$

Additionally, the following terms can also be defined:

$$\mathbf{W}_M = \exp\left(\frac{+j2\pi}{M}\right) \tag{2.79}$$

$$\mathbf{W}_N = \exp\left(\frac{+j2\pi}{N}\right) \tag{2.80}$$

$$\mathbf{W}_P = \exp\left(\frac{+j2\pi}{P}\right) \tag{2.81}$$

Using equations 2.73 to 2.81 , it is then possible to re-express the far-field beam pattern (equation 2.68) as a DFT. The following equation shows the result:

$$D(f,f_x,f_y,f_z) = \sum_{m=-M'}^{M'} \sum_{n=-N'}^{N'} \sum_{p=-P'}^{P'} c_{mnp}\, W_M^{qm} W_N^{rn} W_P^{sp} \quad (2.82)$$

where q, r, and s are defined in equations 2.73 to 2.75 All phasing and steering is done via the complex weights Cmnp.

The problem seems to be solved. The idea is simply to use enough elements in the volume array to allow the representation of the conformal array, and then to set all the elements which do not belong to the array being simulated, i.e., those that do not correspond to an actual element, equal to zero (amplitude weight = 0). Figure 2.2 shows a two-dimensional picture of this process.

This procedure may work for evenly spaced elements but it is easy to see that a very large number of elements of the array will be set to zero (amplitude weight = 0). Moreover, if the elements are not evenly spaced, as in the BQR-7E array, a large number of elements will be required in the volume array in order to exactly represent the conformal array. Furthermore, only a part of the BQR-7 conformal array is used for a given listening angle. This requires careful bookkeeping of which elements are indeed activated (or used) and those which are not (amplitude weights temporarily set to zero).

The problem is that the FFT algorithm does not take into account whether an element is actually present or absent. The algorithm will perform the (M log M)(N log N)(P log P) operations necessary to generate the beam pattern no matter how many elements are actually present.

For example, a conformal array of 100 elements may require a large corresponding volumetric array of 15 x 20 x 5 elements where 1400 elements have an amplitude weight of zero. A total of about 1600 operations are

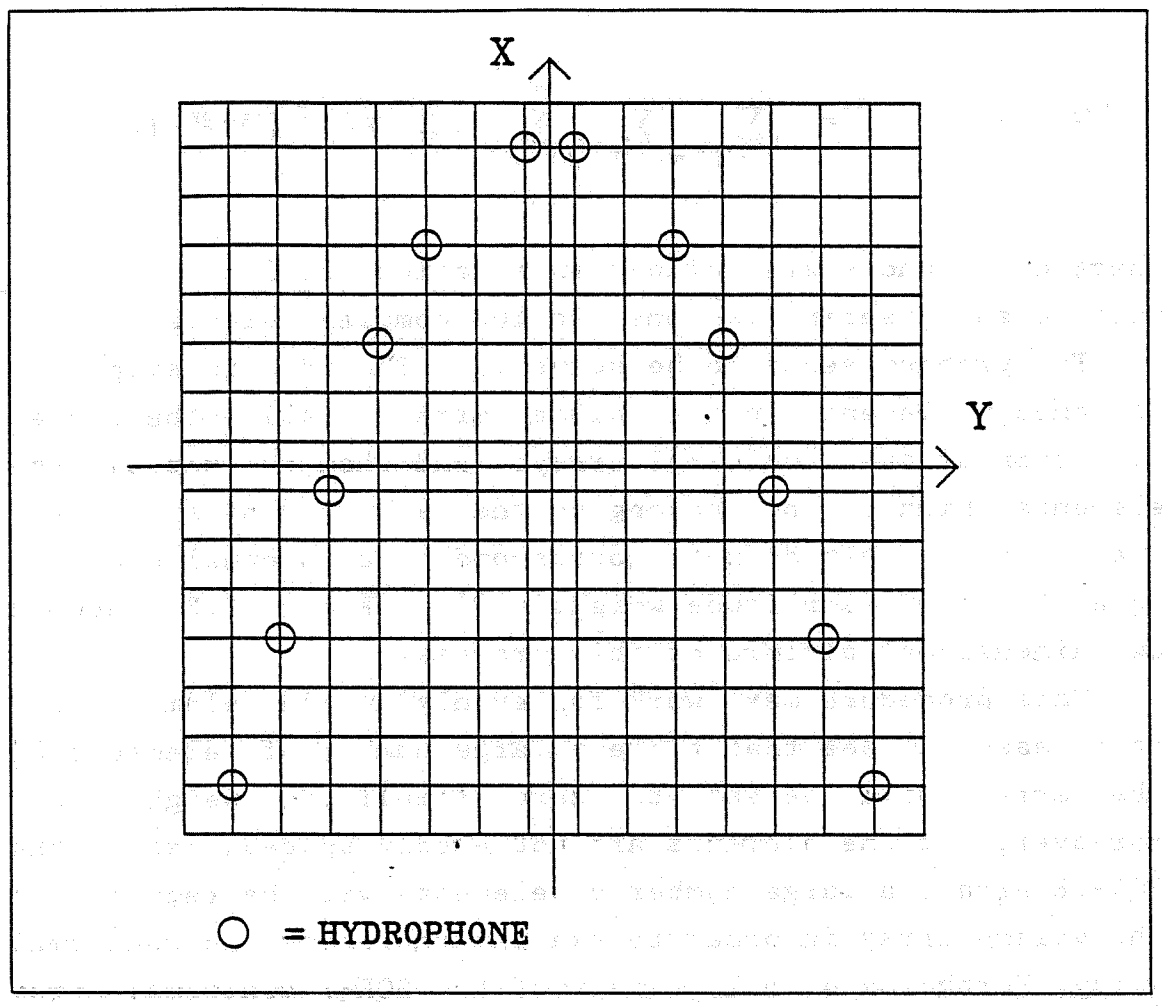

Figure 2.2 Two-dimensional Array Mapping.

performed by the FFT to produce the beam pattern. If a "power of 2" FFT algorithm is used, the array becomes a 16 x 32 x 8 array padded with more zeros, which will require a total of about 6700 operations. The resulting beam pattern will be coarse since only the view angles were used (the sampled ones). In order to refine the beam pattern curve more padding can be done (as for the line array). Thus, an array of 256 x 256 x 64 elements can be used, bringing the approximate total operations performed by the FFT to 44 million. This is a lot of calculations for a decent beam pattern, i.e., a data point for about every

degree in the horizontal and the vertical. This would take many hours of micro-computer time even excluding the book-keeping for the time delays used, the amplitude windows, etc.

Therefore, another approach must be used in order to reduce the amount of calculations while producing an accurate and detailed far-field beam pattern. This new approach is probably adequate for an array containing a large number of elements with some symmetry, and where only a coarse beam pattern curve is required.

F. THE CONFORMAL ARRAY

For the BQR-7E array only a relatively small number of elements is used at a given time, about 80 elements or hydrophones. The plan is to map the individual element to the fixed center of the entire conformal array and to disregard any symmetry considerations, i.e., unequally spaced complex-weighted point source elements.

It will be shown that the technique described here will be more elegant, more adaptable, and the software will be simpler to maintain. It will also simplify the incorporation of the variation of the amplitude weights as a function of the horizontal and vertical angle due to the diffraction of the signal on the body of the submarine. This will be discussed later.

Starting with the expression of the far-field beam pattern for complex-weighted point sources, equations 2.67 and 2.63 can be used to define equation 2.83

$$D(f, f_x, f_y, f_z) = \mathbf{F}_x \, \mathbf{F}_y \, \mathbf{F}_z [s(x,y,z)] \qquad (2.83)$$

If at a later time the actual beam pattern of the individual hydrophones can be made available, the product theorem (equation 2.62) can be used to compute the far-field beam pattern.

Each element can be localized by its actual position, so an element i has the position (xi, yi, zi). Therefore the complex frequency response or aperture function s(x, y, z) can be written (similar to equation 2.60) as:

$$s(x,y,z) = \sum_{i}^{\left[\substack{All \\ elements}\right]} c_i \; \delta(x-x_i)\delta(y-y_i)\delta(z-z_i) \qquad (2.84)$$

Using equation 2.83, this then becomes:

$$D(f,f_x,f_y,f_z) = \sum_{i}^{\left[\substack{All \\ elements}\right]} c_i \; exp\,[+j\;2\pi(f_x x_i + f_y y_i + f_z z_i)] \qquad (2.85)$$

where

$$c_i = a_i \; exp\,(+j\theta_i)$$

As usual θ_i can be modified to steer the array, and a_i can be modified to shade the array. However, since complex sensitivity will be used, θ_i will represent the phase of the sensitivity and any time delay will be added to it, i.e.,

$$\theta_i \rightarrow \theta_i + 2\pi f \tau_i \qquad (2.86)$$

where τ_i = *time delay of element i*

θ_i = *phase of the complex weights of element i*

Moreover, the amplitude weights can be expressed as the amplitude of the sensitivity times any amplitude window coefficients, i.e.,

$$\rightarrow \quad a_i \; b_i \qquad (2.87)$$

where a_i = *Amplitude of sensitivity of element i*

b_i = *Amplitude of window applied to element i*

If the amplitude window is a complex window, then equation 2.86 will contain an additional term, the phase of the window.

Thus the far-field beam pattern can be expressed by equation 2.88 .

$$D(f,f_x,f_y,f_z) = \sum_{i}^{\left[\substack{All\\elements}\right]} a_i\, b_i \qquad (2.88)$$
$$\exp\left\{+j\left[\theta_i + 2\pi f\,\tau_i + \phi_i + 2\pi\left(\frac{\sin\theta\cos\psi x_i + \sin\theta\cos\psi y_i + \cos\theta\, z_i}{\lambda}\right)\right]\right\}$$

where ϕ_i = *Phase of window applied to element i*

Basically, the three dimensional far-field beam pattern is computed using the actual location of the elements, their complex weights, the amplitude window, and their time delays for both the horizontal and vertical angles of direction.

Equation 2.88 will be the basis for the algorithm to be used in this thesis. However, the sum of i applies only to the elements used for a particular case (i.e., at a specific listening angle). Additionally, subscripts can be used for bookkeeping as long as all the relevant elements are considered.

The next section will present some typical cases.

G. TYPICAL CASES

1. Beam Pattern for the Horizontal Angle Only (same x and y)

One application of the conformal array approach is the case of a submarine's array formed of vertically-oriented line arrays of three hydrophones each, called staves, with the outputs of each hydrophone in the stave

connected in parallel. The directivity in the vertical direction is ignored so that only the horizontal angles are considered. The stave positions are given in terms of coordinates x, and y. Then the far-field beam pattern D can be expressed by equation 2.89 .

$$D(f,f_x,f_y,f_z) = \sum_{m}^{\left(\substack{All\\staves}\right)} \sum_{p}^{\left(\substack{3\\elements}\right)} a_{mp}\ b_{mp}$$
$$\exp\left\{+j\left[\theta_{mp} + 2\pi f\,\tau_{mp} + \phi_{mp} + 2\pi\left(\frac{\cos\psi\ x_m + \sin\psi\ y_m}{\lambda}\right)\right]\right\} \qquad (2.89)$$

Since sin θ = 1 and cos θ = 0 at 90 degrees.

Here m is the stave number and p is the hydrophone number in a stave. The portion affecting the z dependence is eliminated, except for the amplitude weights a .

For a fixed vertical angle of 90 degrees, the beam pattern D becomes

$$D(f,\theta,\psi) = D(f,\psi) \qquad (2.90)$$

Moreover, as will be described in the next chapter, the time delay, required to compensate for the curvature of the array to simulate reception by a planar array aligned perpendicular to the listening angle, is applied to an entire stave (which is formed of three elements). There is no dependency on Z, so:

$$\tau_{mp} \rightarrow \tau_m \qquad (2.91)$$

The same applies to the amplitude window, where

$$b_{mp} \rightarrow b_m \tag{2.92}$$

and

$$\phi_{mp} \rightarrow \phi_m \tag{2.93}$$

Figures 2.3 and 2.4 show the arrangement.

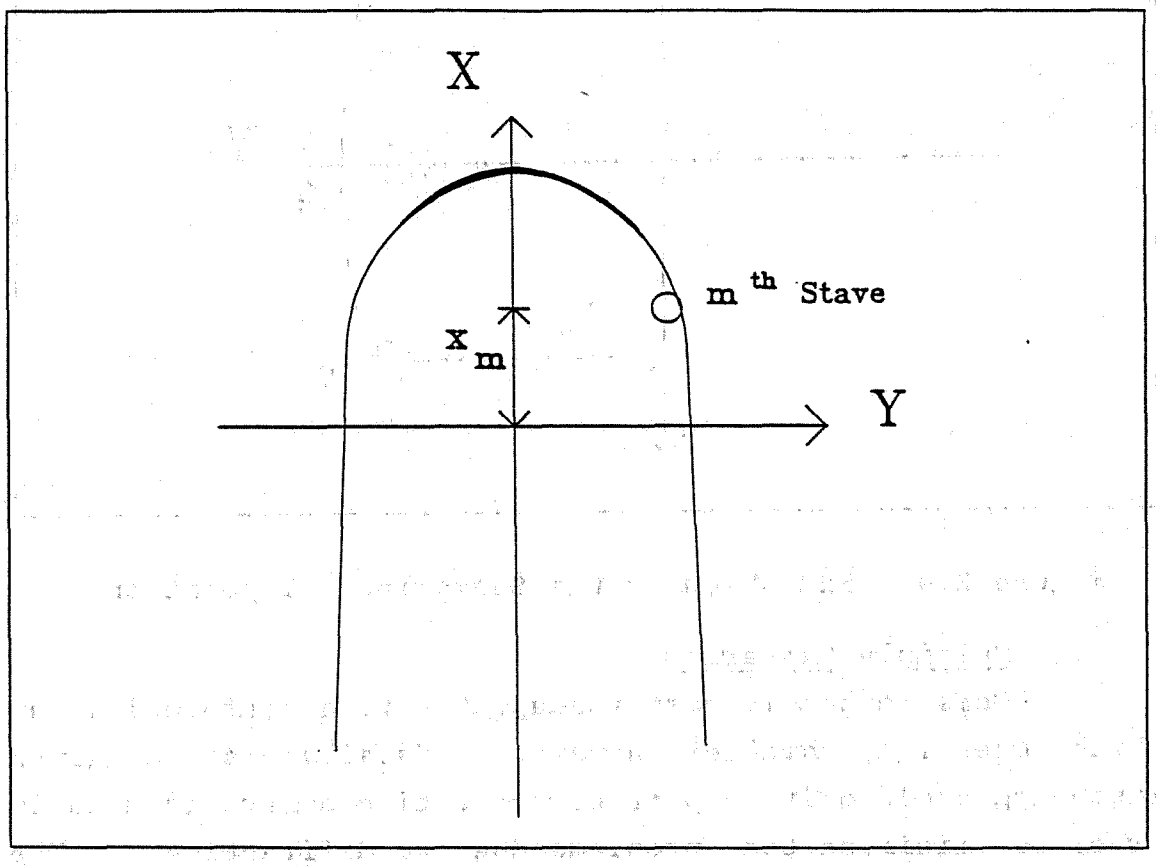

Figure 2.3 Plan View of the Submarine - X position.

2. Beam Pattern for the Horizontal Angle Only (different x & y)

Another case with different x and y positions for each element with the same z position, i.e., all elements on the top array are the same vertical distance from the next one below is provided.

The mth stave is at position xm, and the y position of the hydrophones in the stave is ymp in equation 2.89 . Figures 2.5 and 2.6 show this arrangement.

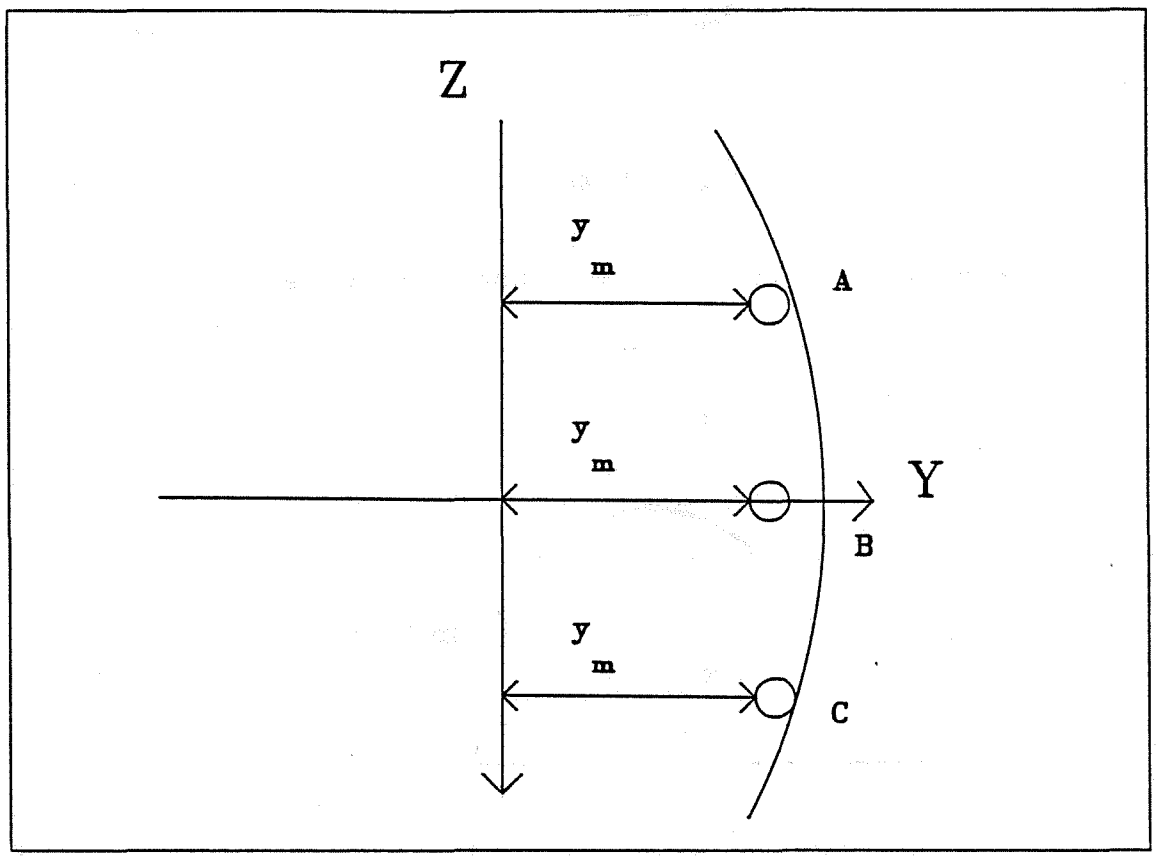

Figure 2.4 Side View of the Submarine - Y position.

3. Multiple Sub-arrays

This method is easily adaptable to a conformal array which uses many smaller arrays. Usually, at a given listening angle only a part of the entire array is used in order to minimize the problems due to diffraction. The array is further subdivided to receive the appropriate positive and negative time delay, and to be filtered through an amplitude window.

As an example, for a listening angle of 169 degrees in the BQR-7, three sub-arrays are used: a primary and secondary right sub-array and a primary left sub-array. The far-field beam pattern simply becomes equation 2.94 using the same x, y as for a stave;

$$D(f,\psi) = \left(\sum_{(m=41)}^{48} + \sum_{(m=49)}^{52} + \sum_{(m=1)}^{4} \right) \sum_{(p=-1)}^{1} a_{mp}\ b_m \exp\left\{ +j \left(\theta_{mp} + 2\pi f \tau_m + \phi_m + \frac{2\pi}{\lambda} \left(\cos\psi\ x_m + \sin\psi\ y_m \right) \right) \right\} \quad (2.94)$$

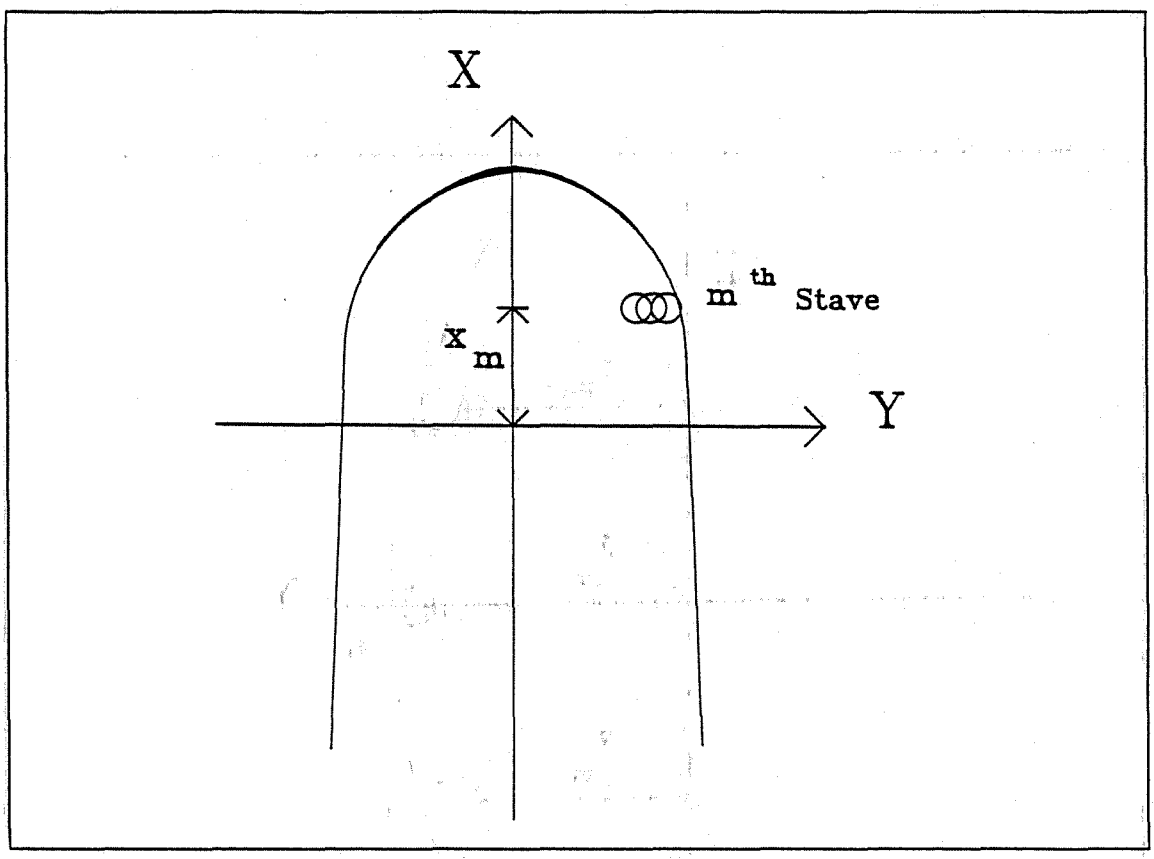

Figure 2.5 Plan View of the Submarine - X Position.

4. Frequency Dependence of the Complex Weights

Usually, a normal transducer used as a receiver has a flat frequency response for the range of interest, say up to 3 kHz. This is the basis for the usual generalization that the complex weights are constants, independent of frequency. But for a degraded hydrophone, the response may not be flat so that a dependency for the complex weights on the frequency should be included.

So

$$a_{mp} \rightarrow a_{mp}(f) \qquad (2.95)$$

and

$$\theta_{mp} \rightarrow \theta_{mp}(f) \qquad (2.96)$$

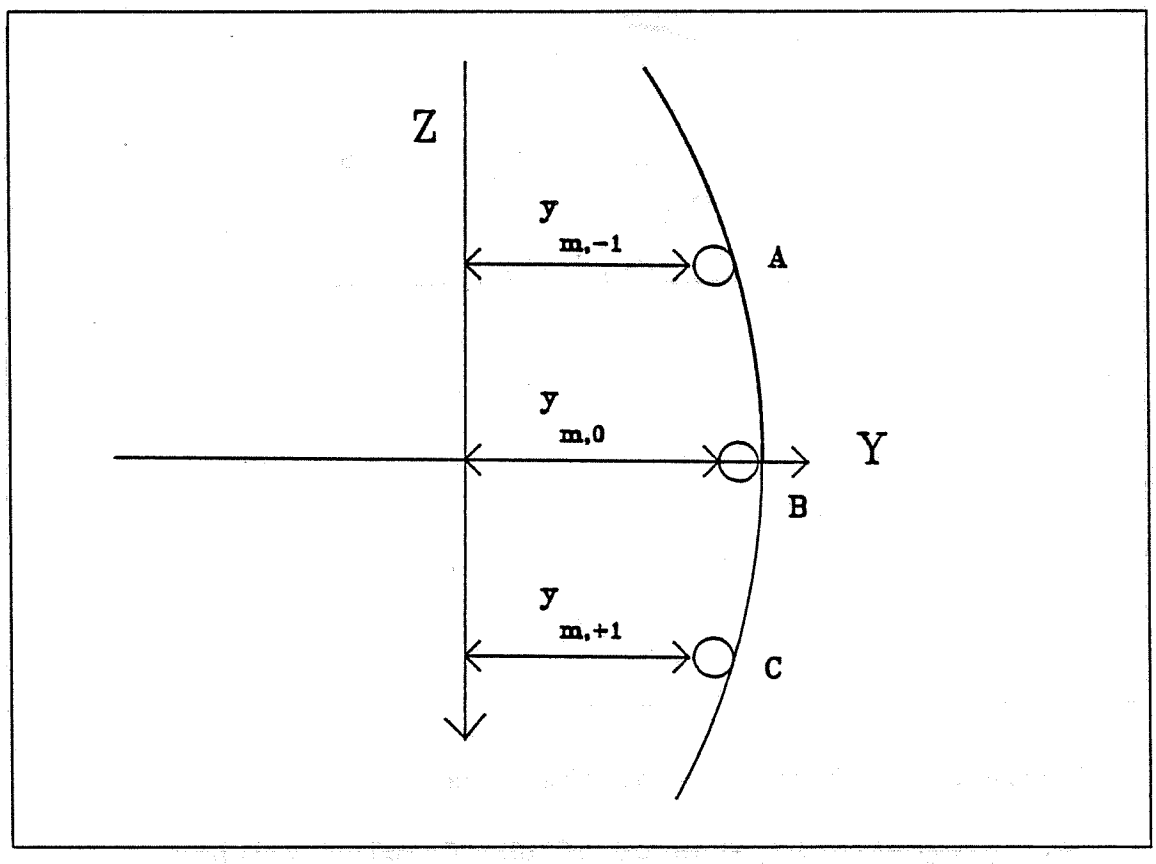

Figure 2.6 Side View of the Submarine - Y position.

Therefore, equations 2.89 to 2.96 can be combined and then divided into two portions: a real part and an imaginary part of the Far-Field Beam Pattern. They are expressed by equations 2.97 and 2.98 .

$$Re\,(D\,(f\,,\psi)) = \sum_{m}^{\left(\substack{All\\staves}\right)} \sum_{p=-1}^{1} a_{mp}\,(f\,)\; b_m\; \cos(\Omega_{mp}) \qquad (2.97)$$

$$Im\,(D\,(f\,,\psi)) = \sum_{m}^{\left(\substack{All\\staves}\right)} \sum_{p=-1}^{1} a_{mp}\,(f\,)\; b_m\; \sin(\Omega_{mp}) \qquad (2.98)$$

where

$$\Omega_{mp} = \theta_{mp}\,(f\,) + 2\pi f\,\tau_m + \phi_m + \frac{2\pi}{\lambda}\left(\cos\psi\; x_m + \sin\psi\; y_m\right) \qquad (2.99)$$

These will be the equations to be used in the software algorithm of Chapter 3.

H. HYDROPHONE SENSITIVITY

The element's complex sensitivity is an expression of the transfer function of the output versus the input as a function of frequency. Here it is the ratio of an element's electrical output to a specified load (such as open current voltage, short circuit current, etc.) to the applied acoustic pressure (see [Ref. 1: Chap2] and [Ref. 4: p. 353]) at a given frequency.

Normally, sensitivity is given by the complex (magnitude and phase), free-field, open-circuit sensitivity level of the hydrophone expressed in decibels relative to 1 volt per micropascal. The "sensitivity" of the array which is determined by the sensitivities of all the individual elements, forms the complex aperture function. It is also called the

complex frequency response or the free-field voltage sensitivity.

It is important to mention that only the relative complex sensitivities are required, not the absolute ones. The expression of the difference between each element is used in order to generate the Far-Field Beam Pattern and to determine its degradation.

Finally, according to Self [Ref. 7], the uncertainty of the calibration measurement varies from +/- 0.25 dB for the static case to about +/- 1.0 dB, which is very acceptable for the evaluation of the beam pattern.

I. DIFFRACTION DUE TO THE HULL

When sound waves meet an obstacle, such as a submarine's hull, they will spread around the edges of the obstacle giving rise to diffraction of sound. If the hull surface is relatively transparent acoustically, the internal components such as the bulkheads, tanks, and other equipments, will also cause diffraction effects. Additionally, sound waves will be scattered in all directions when they strike obstacles of dimensions small compared with their wavelengths. The larger the ratio of the wavelength to the dimensions of the obstacle the greater the diffraction.

The diffraction of sound by a simple object such as a sphere, cube, or cylinder has been known for a long time. Figure 2.7 [Ref. 8: p. 6] shows the diffraction of sound by a sphere, a cube, and a cylinder as a function of the dimensions [Ref. 9: p. 20]. The ratio of pressure over the static pressure in dB indicates that for a cylinder no change in pressure is measurable at 180 degrees (behind the cylinder) while 10 dB must be added at 0 degrees when the diameter is equal to the wavelength. A simple extension to the submarine would probably show that the diffraction will play an important role in the measurement of the pressure by localized hydrophones. To develop the approximate

expression for the diffraction of sound caused by the "nose" of a submarine is not a trivial task. Time limitations did not allow the development and implementation of the submarine's diffraction. It is suggested that a thesis be done on this matter with experimental verification. This will later be incorporated into the algorithm to compute the far-field beam pattern.

If diffraction effects are to be calculated, the amplitude weights can be changed by passing them through a 'counter-diffractive' window which will adjust the amplitude weight in order to reflect the correct measured data. It is important to mention that the window is a function of the horizontal angle. For a new angle a new window is created and then is applied to the proper sub-array.

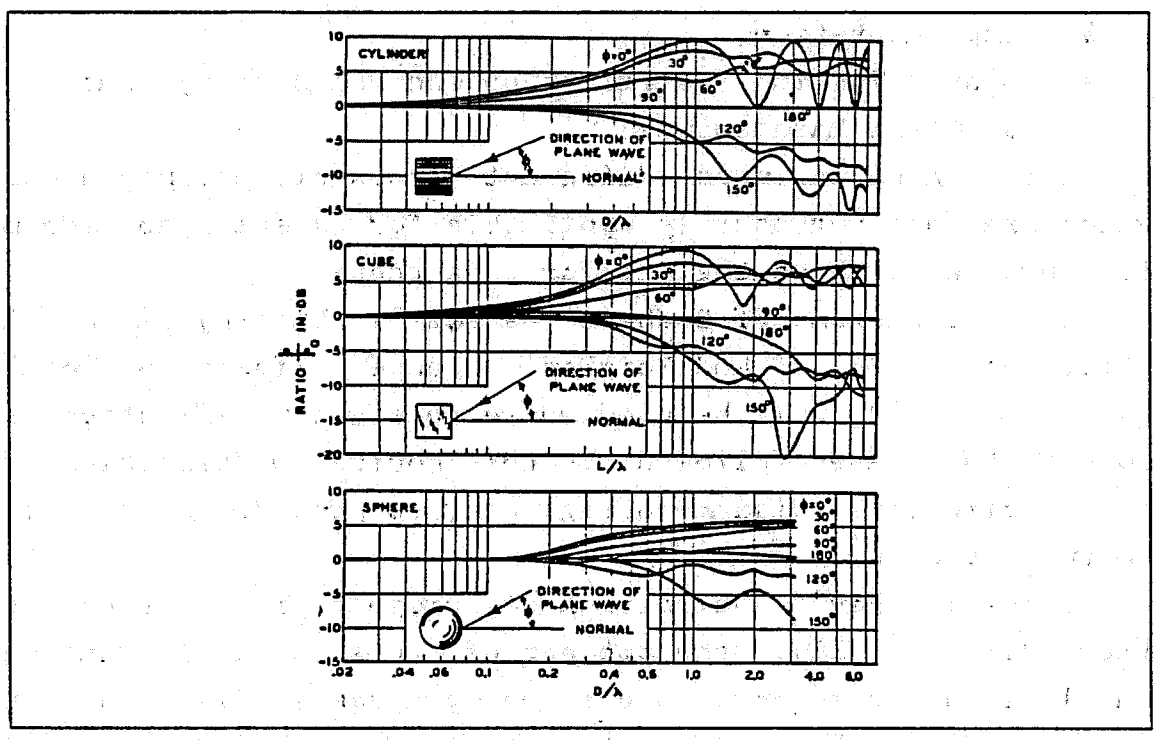

Figure 2.7 Diffraction due to Simple Object.

III. ANALYSIS, DESIGN, & IMPLEMENTATION

A. THE PHASES OF A SOFTWARE PROJECT

Software engineering can provide many details on how to successfully and systematically design, produce, and maintain a software product or a software system. There are two main stages, software development, addressed extensively by this thesis, and software maintenance, which will be rapidly covered in the form of proposed techniques. The basic life-cycle model of software development and maintenance can be separated into five phases. These phases are:

1. Analysis (and Planning);
2. Design;
3. Implementation;
4. System Testing (Verification and Validation); and
5. Software Maintenance.

Each phase includes different types of documentation which are not necessarily applicable for a small-to-medium isze program such as the one for this thesis.

A combination of Boehm's Waterfall Life-cycle Model [Ref. 10: p. 36], and Fairley's Phased Life-cycle Model [Ref. 11: p. 42] will be used where constant verification is done on the phases followed by the required modification of the requirements, design coding, etc, to produce an adequate software product.

Using Fairley's definition of project size categories [Ref. 11: p. 11], the project of this thesis is in the small-medium category, where one programmer is used for about 9 months to produce two to four thousand lines of source code (software program).

The analysis phase defines the problem to be solved and what the goals are. If a computerized solution is justified, then a solution strategy is adopted after a

feasibility study of each possible strategy. Next, a life-cycle model is defined and the actual development process is initiated. Basic software requirements for the product are also defined. This is an extremely important phase for medium and large software projects.

In the design phase, a specific design technique is used to identify the software components (functions, data streams, and data stores), and specify the software structure. The design consists of the external, architectural, and detailed design. The results are written as design specifications using data flow diagrams, control flow diagrams, and structured charts.

The implementation phase of software development involves translation of the previous design specifications into source code (the actual software program), and then debugging, documenting, and unit testing the source code.

The next phase, system testing, is divided into integration testing and acceptance testing. These validate and verify the proper functioning of the software product and its software components. This means that the product is verified (Does it work? Is the product right?), and then validated (the acceptance test which answers the question: Is it the right product?). No validation will be performed by this thesis since no data were made available.

Finally, the software product (system) is released for operational use and enters the maintenance phase of its life-cycle management. Maintenance activities include enhancement of capabilities, adaptation of the software to new processing or operational environments, and correction of any software errors and performance deficiencies.

B. ANALYSIS : DEFINING THE PROBLEM

1. The Problem

Chapter one described the reasons for developing a simulation and modeling software program for a conformal array. In an operational environment the degradation of the elements of the array affects the beam pattern. It is important for tactical reasons to know the specific shape of this degraded beam pattern. First, one must know the shape of the non-degraded conformal array beam pattern (as per design). This is the modeling portion. Then the actual conformal array can be simulated.

With the recent developments in instrumentation and digital computers, it is now possible to produce the beam pattern of a degraded conformal array accurately and rapidly. The submarine commander, or the operator of the SONAR array, can then have an accurate estimate of the status of the array performance, such as the new range, listening angle variations, beamwidth changes, and shape deformation for any arrangements of its hydrophones' degraded sensitivities.

A stand-alone user-friendly software program can be a feasible solution for obtaining quick and accurate far-field beam pattern data for any specific situation, such as the speed of sound, the frequency, the complex sensitivities, array configuration, and mode of operation.

First, it is important to use a specific conformal array shape, such as the type mounted on the hulls of most submarines. The next subsection will describe a specific yet typical array system.

2. The BQR-7E Acoustic Array - A Model

No specific hydrophone will be specified, since it can easily be implemented later using the product theorem as defined by equation 2.62 Therefore, all hydrophones will be considered as complex weighted point sources (i.e., point

sources with a complex relative sensitivity). A conformal array similar to the AN/BQR-7 hydrophone array will be used. Any similar shape, even the one of a surface ship, can be used with the proper computing adjustment. For a surface ship, for example, additional assumptions must be made due to the surface scattering and the variation of the speed of sound in the mixed layer as function of depth [Ref. 12].

A total of 156 hydrophones, 52 groups (defined as staves) of three hydrophones each, are arranged in a configuration fitted to the shape of the bow of the ship. Nine hydrophones are missing on the model due to the location of torpedo tubes, antennas, etc. Figure 3.1 shows the configuration and identification number (1 to 52) of each stave of the array.

Normally, the three hydrophones in a stave are connected in parallel at a junction box. The hydrophones of a stave are labelled A (Top), B (Center), and C (Bottom). An alternative notation is -1, 0, and +1 (relative to the direction of the Z axis (going into the page)). Figure 3.2 shows a stave and its labels.

As indicated earlier, a few hydrophones have been omitted from the array because of obstructions in the hull. For the system considered here, the following hydrophones are missing:

1. No. 25 A (-1) and no. 25 C (+1);
2. No. 26 A (-1) and no. 26 C (+1);
3. No. 27 A (-1) and no. 27 C (+1);
4. No. 28 A (-1) and no. 28 C (+1); and
5. No. 21 B (0).

Again, a group of three hydrophones is referred to as a stave.

Each stave is connected to a preamplifier unit (for a total of 52) which is a sealed, nonrepairable unit with a nonadjustable gain of about 70 dB at 1.5 kHz. Since array

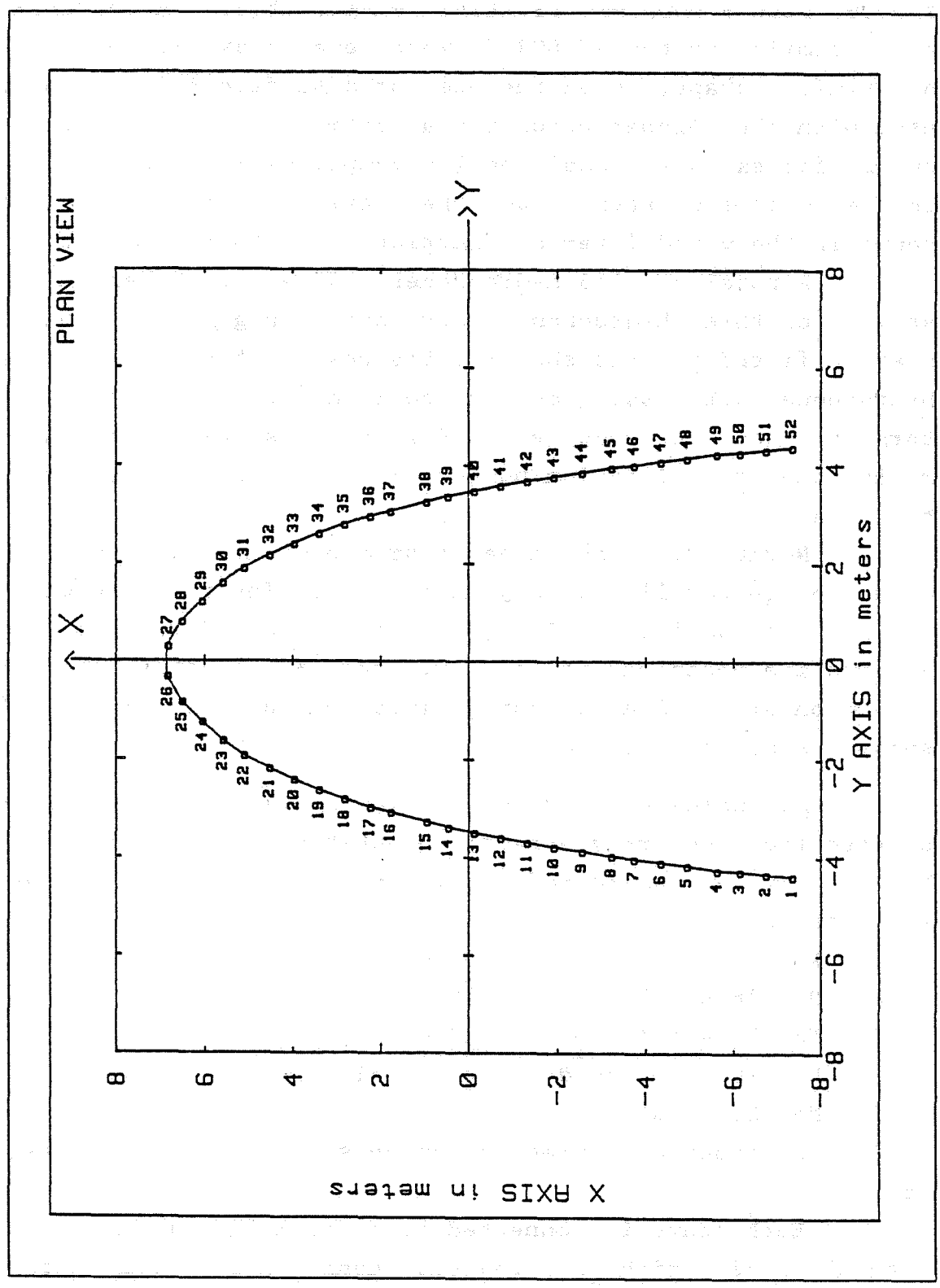

Figure 3.1 Array Configuration - Top View.

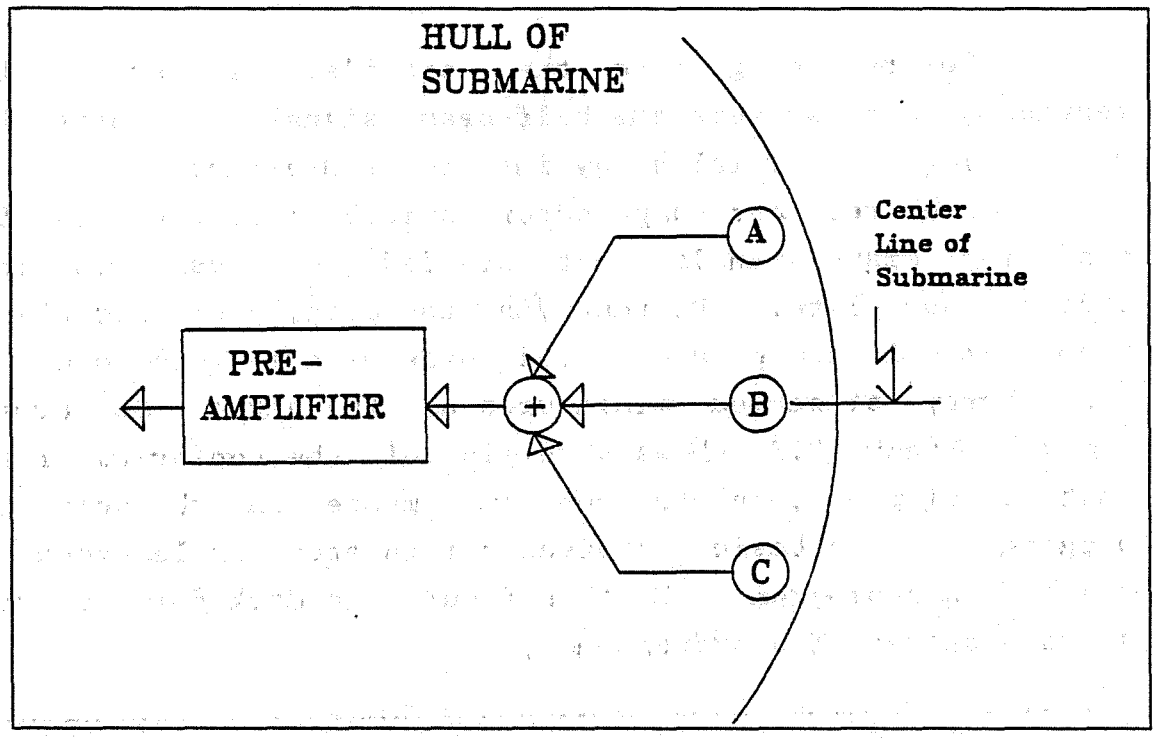

Figure 3.2 A 3-Hydrophone Stave.

theory requires all preamplifier channels to produce signals of virtually the same amplitude and phase shift in order to obtain optimum bearing accuracy, all preamplifiers will be considered to be identical. Their relative effects on the shape of the beam pattern are therefore eliminated. Future versions of the software program may contain a feature which will express the actual amplitude and phase shift of each preamplifier, eliminating unnecessary replacement of these units.

The preamplifier signals from the 52 staves are fed to a beamforming device which is a recorder (automatic scanner) and a manual compensator electronic switch. This device simulates the rotation of the array to a specific listening angle. The beamforming circuits select which stave/preamplifier signals are to make up the listening beam by positioning a compensator switch.

Before going into the details, it should be mentioned that two separate half-beam signals are formed by subdividing the partial array into two sub-arrays.

The recorder compensator switch thus receives 52 amplified stave signals that are fed to brushes on the stator brush plate. Disregarding the details of this electromechanical compensator, the brushes are located on a model array of scaled miniatures of the actual conformal array. Figure 3.3 shows a version of the conformal array that is simpler than the one implemented in the software program and its basic conformal compensator implemented in the software program. In this figure, a dark point represents a stave (of 3 hydrophones).

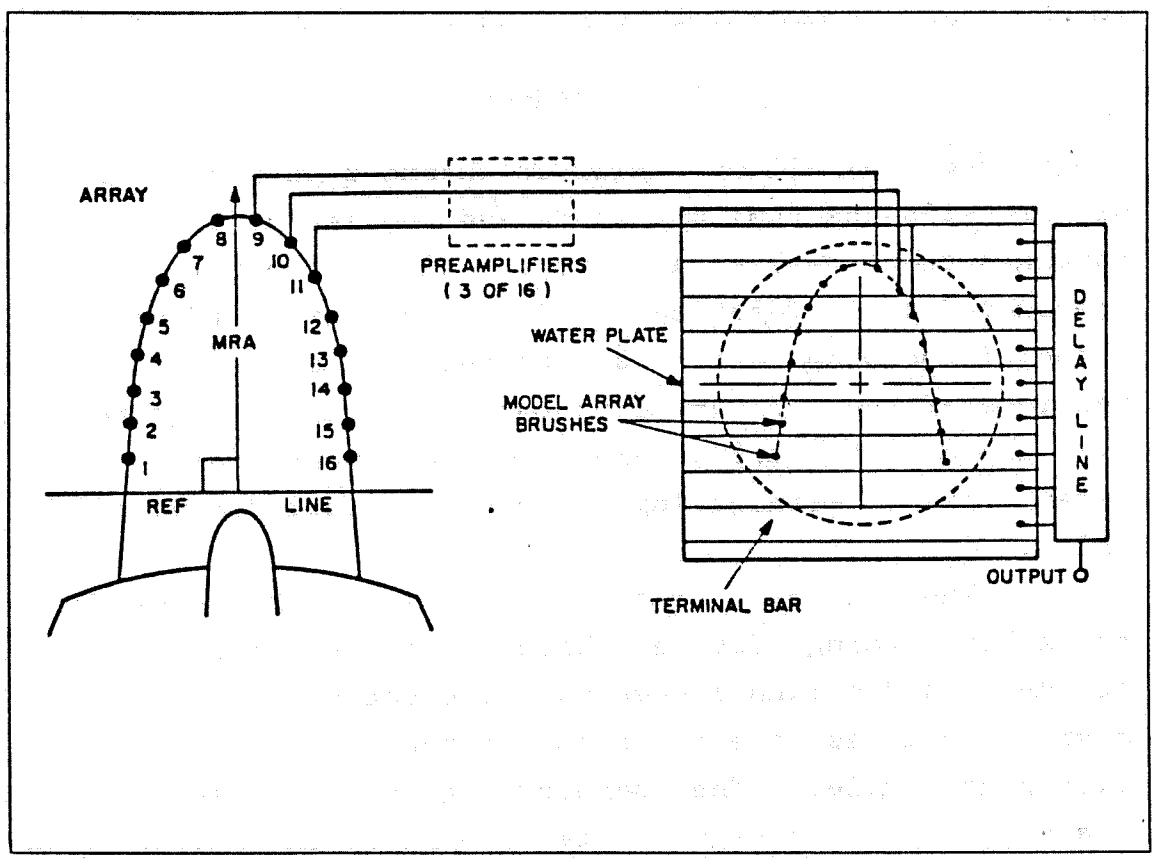

Figure 3.3 Conformal Array and Basic Conformal Compensator.

The lower plate, called the water plate or compensator plate, has a dual function. First, it selects which brushes, and therefore which staves, are to be used to form the listening beam. Secondly, it provides the switching that supplies the proper amount of time delay to the signal from each of the selected staves to compensate for the curvature of the array (refer to Chap. 2). This simulates reception by a planar (flat) array aligned perpendicularly to the listening angle. In the case of the simulated array, two time delay lines are used (instead of one as shown in Figure 3.3), one with positive time delays, and the other with negative ones. Additionally, two model array brushes are used, one for each line. Figure 3.4 shows the water plate used for the actual array.

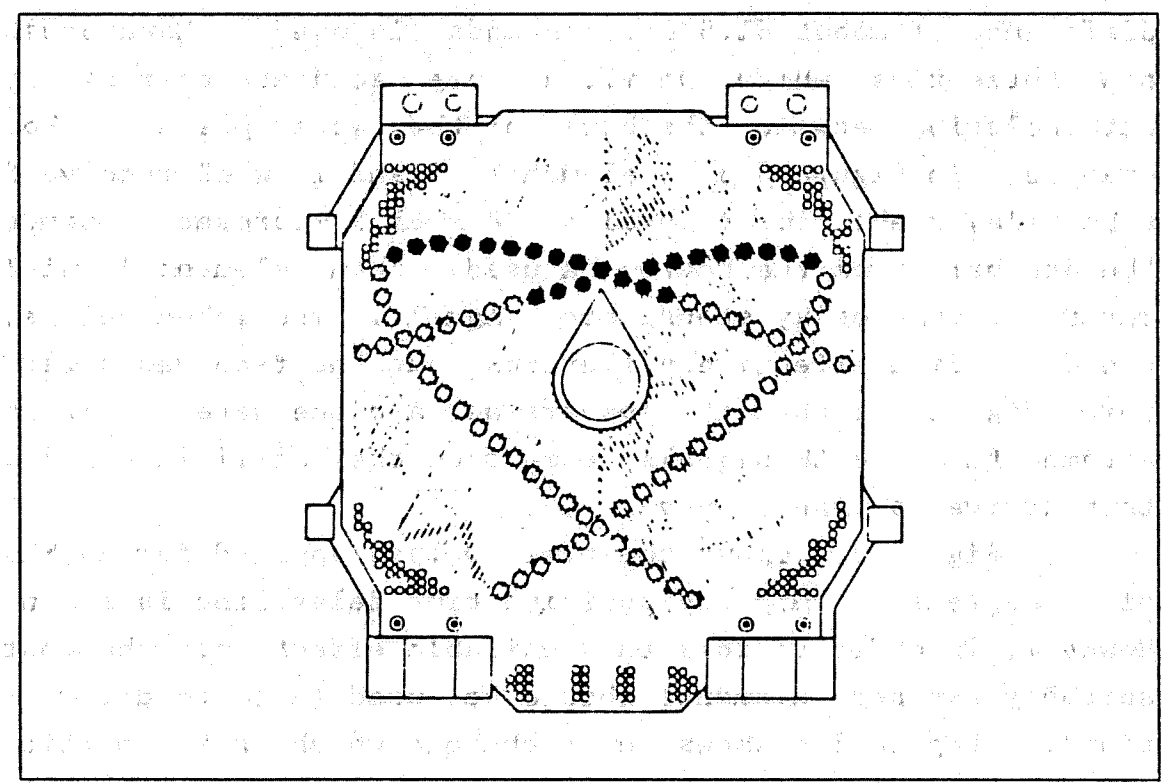

Figure 3.4 Compensator Plate with Two Model Arrays.

Thus some signals go to the left delay line, some to the right, and some are grounded. The effect of this action is to create two effective arrays called the left and right sub-arrays. Up to 14 stave signals for the left sub-array, and 14 for the right sub-array can be used for a selection of 28 of the 52 staves depending on the listening angle. The listening angle is also called the maximum response axis (MRA). The model array can be rotated to an MRA either automatically or manually by an operator. The result of stave selection and time delays is to create two effective sub-arrays in a straight base line some short distance behind the true array.

Referring to Figure 3.3 and its time delay, note that time delays between the terminal bars are discrete. A difference of about 61.5 microseconds is used. Each brush has three tips which permit a more accurate bearing by interpolating between the bars of the water plate. For example, in Figure 3.3 , elements 9 and 10 will receive 8 time delay steps for a total of 8 x 61.5 microsec., since the 8th bar from the bottom is used, then element 11 will receive 7 time delay steps, etc. Again as mentioned before, the delay line effectively flattens out the received water-borne signal so that it appears as a plane wave. It is assumed that the CW signals come from the far-field, i.e., they arrive as plane waves.

Figure 3.5 shows the time delays applied for an MRA of 45 degrees. Only the positive time delay line is shown. However, in order to form the desirable effect only the most suitably exposed elements should be used (due to diffraction). Figure 3.6 shows one technique which uses a portion of the water plate for time delay.

This is basically the method used in the software program. Another method that is also available uses concentric sliprings plus a strator and a rotor brush plate,

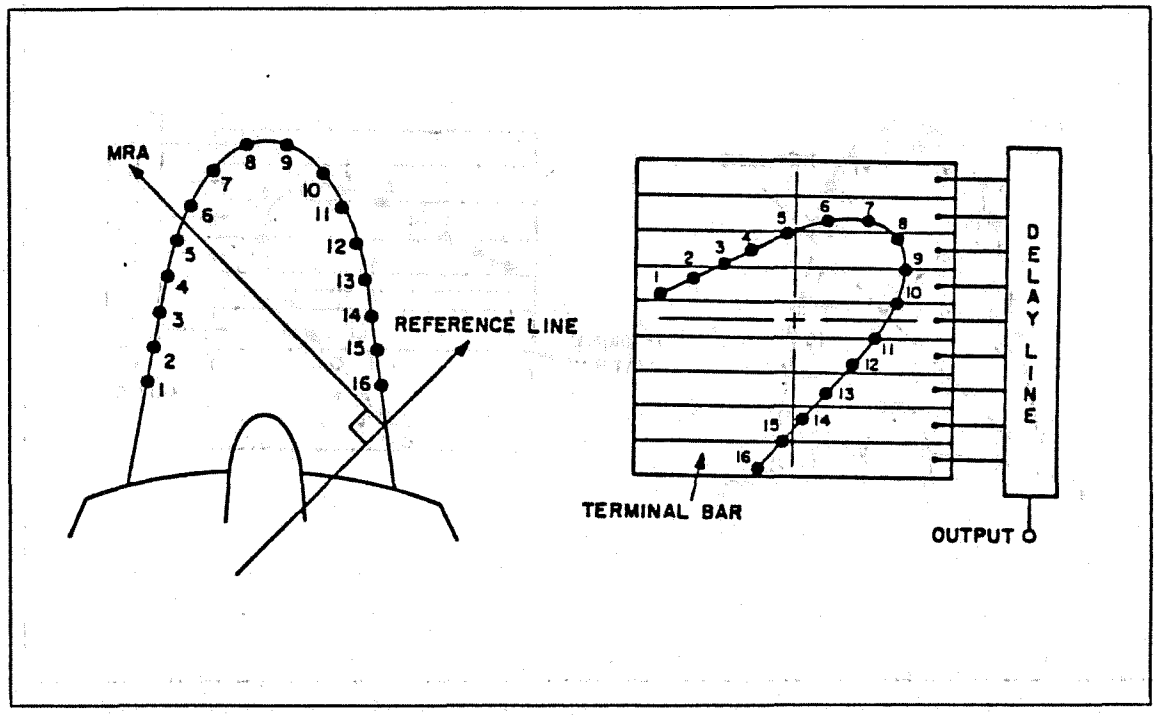

Figure 3.5 Bearing Selection of 45 degrees.

and a compensator stator. Finally, the entire electromechanical device can easily and reliably be implemented using digital methods as is done for many similar devices.

Table I shows the stave selection made for a given MRA. For angles of 166 to 169 degrees, a secondary right sub-array is also used. The most general model uses primary and secondary left and right sub-arrays (not applicable for this thesis). For any negative MRA, the stave number is mapped around the X axis, e.g., 1 becomes 52, 2 becomes 51, etc.

The technical manual describing the evolution of the signal in the beamformer indicates that more processing is done in the BQR-7 system, such as the evaluation of the sea noise, additional filtering, and phase shifting, phase detection, etc. These are not considered here.

One more feature will be considered: the phase shifter. In order to obtain the exact bearing of a target,

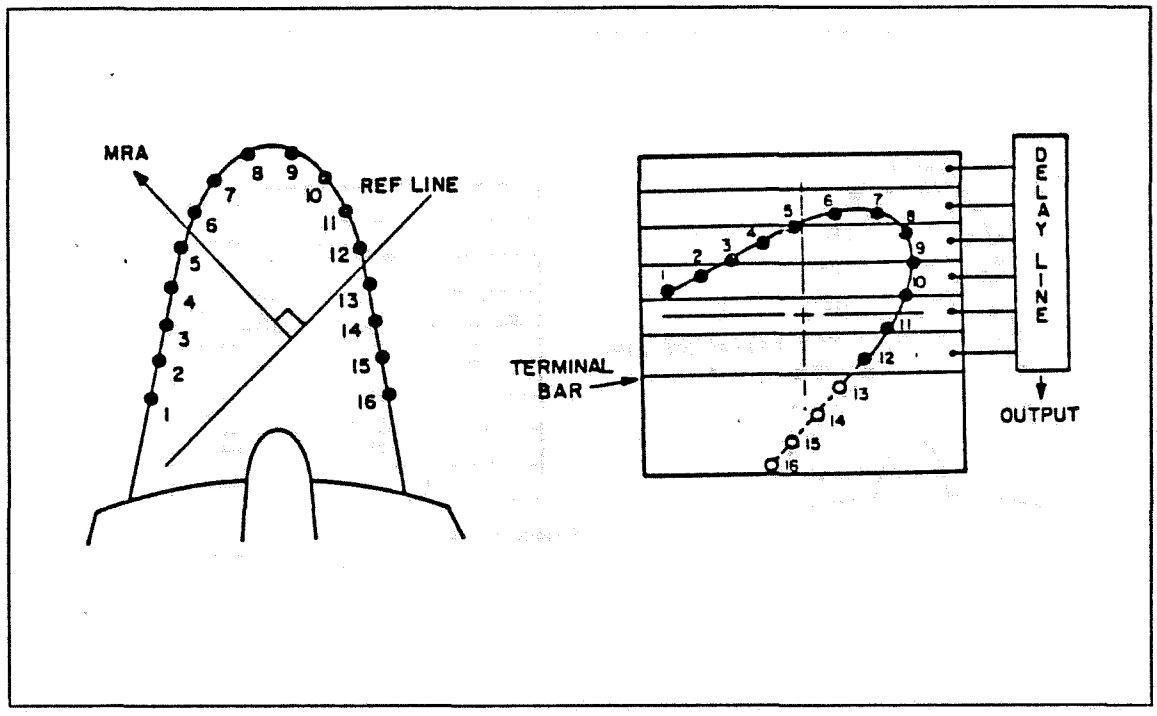

Figure 3.6 Stave Selection Using the Grounding Method.

the detection scheme of the beamformer requires that a maximum signal be developed by the phase detector when the MRA is squarely "on target". This is accomplished in the phase shifter by separating the phase of the incoming signal by exactly 90 degrees. The original signal and the shifted signal are analysed by one of two software routines, depedding on the operator's choice. The rectangular window, called "SUM"; creates a beam pattern by summing the left and right sub-arrays. The other window is a "SIGNUM" (half positive and half negative) window called "DIFF", which forms a new beam pattern by subtracting the left sub-array from the right sub-array. The result is that a maximum with the SUM mode corresponds to a minimum with the DIFF mode which helps to exactly identify the MRA of a target. This effect has also been used with RADAR.

Basically, the far-field beam pattern is generated for a selected MRA using the appropriate sub-arrays, with

TABLE I

STAVE SELECTION FOR A GIVEN MRA

MRA (degrees)	LEFT SUB-ARRAY	No.	RIGHT SUB-ARRAY	No.	TOTAL	No.
000	13-26	14	27-40	14	13-40	28
005	16-27	12	28-41	14	16-41	26
010	18-29	12	30-43	14	18-43	26
015	20-30	11	31-44	14	20-44	25
020	21-32	12	33-46	14	21-46	26
025	22-33	12	34-47	14	22-47	26
030	23-34	12	35-48	14	23-48	26
035	23-34	12	35-48	14	23-45	26
040	24-35	12	36-49	14	24-49	26
045	24-36	13	37-50	14	24-50	27
050	25-37	13	38-51	14	25-51	27
055	25-38	14	39-52	14	25-52	28
060	25-38	14	39-52	14	25-52	28
065	25-38	14	39-52	14	25-52	28
070	26-38	13	39-52	14	26-52	27
075	26-39	14	40-52	13	26-52	27
080	27-39	13	40-52	13	27-52	26
085	27-39	13	40-52	13	27-52	26
090	27-39	13	40-52	13	27-52	26
095	27-40	14	41-52	12	27-52	26
100	28-40	13	41-52	12	28-52	25
105	28-40	13	41-52	12	28-52	25
110	28-40	13	41-52	12	28-52	25
115	28-40	13	41-52	12	28-52	25
120	28-41	14	42-52	11	28-52	25
125	29-41	13	42-52	11	29-52	24
130	30-41	12	42-52	11	30-52	23
135	31-42	12	43-52	10	31-52	22
140	32-42	11	43-52	10	32-52	21
145	34-43	10	44-52	9	34-52	19
150	35-44	10	45-52	8	35-52	18
155	35-44	10	45-52	8	35-52	18
160	36-44	9	45-52	8	36-52	17
165	37-45	9	46-52;1	8	37-52	17
166	38-45	8	46-52;1	8	38-52	16
167	40-47	8	48-52;1-3	8	40-52	16
168	41-48	8	49-52;1-4	8	41-52	16
169	43-50	8	51-52;1-6	8	43-52	16
170-180	45-52	8	1-8	8	45-8	16

the appropriate time delays with the proper window as determined by the selected mode of operation. Since no other features will be modeled, the remaining signal processing, such as evaluation of the signal-to-noise ratio, is not treated in this thesis.

After so many pages of description, it may be hard to visualize what is happening. Figure 3.7 shows a schematic diagram of the evaluation of the waterborne signal up to the point where the far-field beam pattern is defined. In Figure 3.7 , the block called the beamformer includes all the other electronics used in dealing with the signal.

The model used here deals only with horizontal listening angle (azimuthal angle), but if it was desirable, the program could be modified to deal with the vertical angles as well. See chapter 2 for the details.

Except for the beamformer, the features of Figure 3.7 should be preset in the beam pattern program. This includes the choices of MRA, frequency, speed of sound, and mode of operation for a given configuration of a conformal array.

Additional functions will include a MENU for selection of MRA, frequency, speed of sound, mode of operation, and formats of output (graphical and tabular); a MENU for the actual medium (printer, plotter, etc) use for output, and a MENU allowing manual input of individual stave conditions (e.g., what is the beam pattern if a certain stave is removed?). The Design Specifications will define the details.

C. ANALYSIS - DEVELOPING A SOLUTION STRATEGY

Following the definition of the problem and the basic features, a solution strategy was developed. Two professors at the Naval Postgraduate School, S.L. Garrett and O.B. Wilson, initially determined that a software program

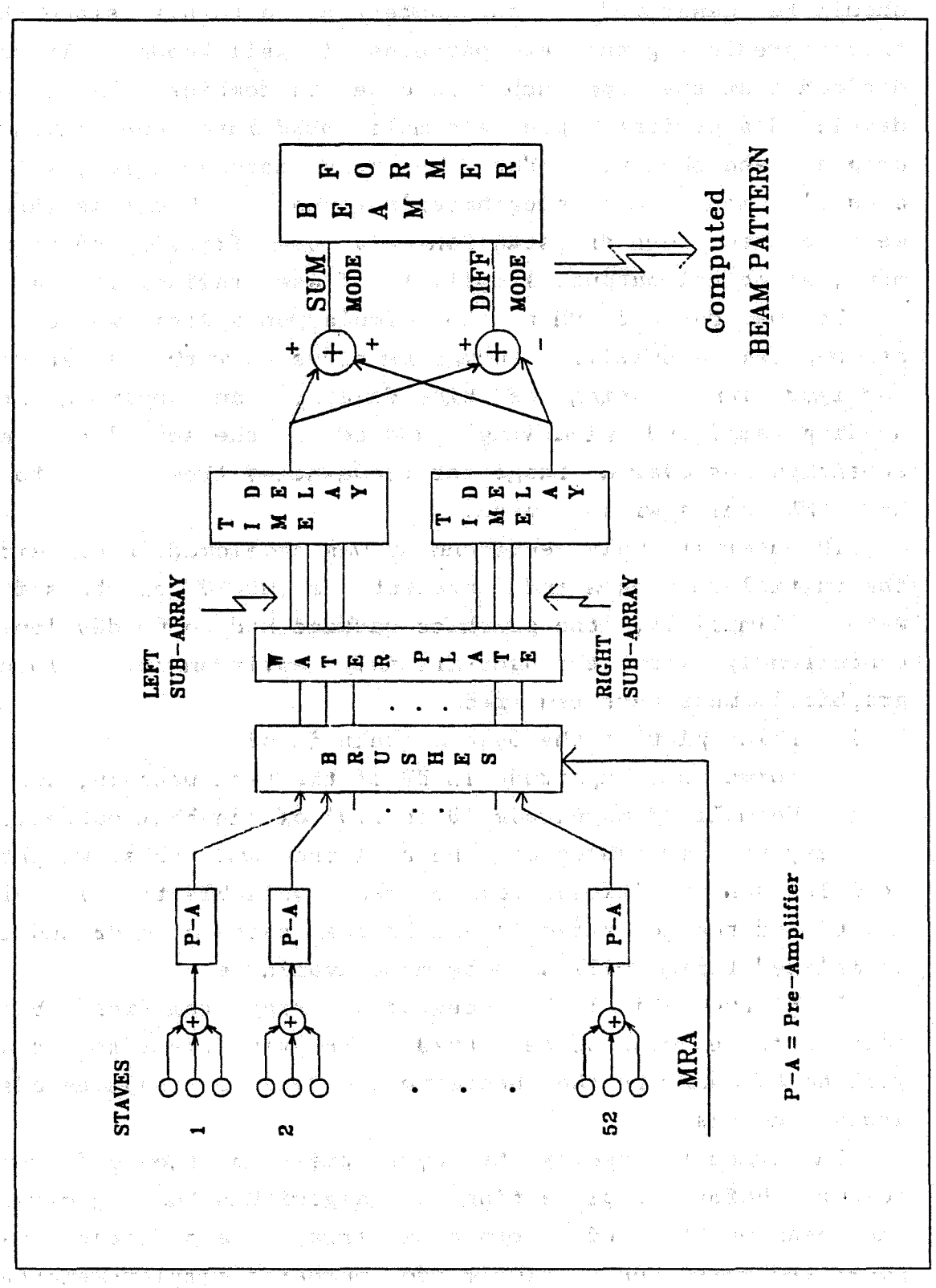

Figure 3.7 Schematic Diagram of Array and Preprocessing.

should be generated in the context of a thesis since the theory predicting the beam patterns is well known. It was decided that the approach should be to combine the to-be-developed algorithms plus a simple MENU into one software program, and then to produce the proper documentation, i.e., a user's manual and programmer's manual. Numerous ideas were adopted such as establishing a user-friendly environment, graphical output, simplicity of use, reliability, etc.

It was decided that the simulation system would be defined for a specific submarine class, with its actual configuration, using its time delays, and applying its shading (amplitude windowing), and to use the actual complex sensitivities over a range for frequencies from 50 Hz to 5 kHz. MKS units will be used.

The user-friendly environment was developed along with the initial data base and represents about 40% of the software. Similarly, the graphics package had to be developed concurrently with the user-friendly environment. Three graphic formats were required:

1. Polar plot of the beam pattern in dB,
2. Normalized magnitude in dB of the beam pattern, and
3. Normalized magnitude (0 to 1.0) of the beam pattern.

Easy to use tables of the amplitude and phase weights (complex sensitivities) will be made available to the user. A table of the generated far-field beam pattern in dB and in normalized format will also be made available.

Both theoretical and degraded array far-field beam pattern curves must be generated. However, the first stage will be to describe the theoretical curves for an undegraded array. curves.

In order to verify the correctness of the different features before implementing the algorithms that generate the beam pattern of a conformal array, a prototype was generated based on a simple odd numbered complex-weighted

point source element array lying along the X axis. This greatly simplified the development stage, and provided data on the performance of the system, its reliability, speed, and accuracy in computation. It will be particularly valuable for obtaining a better understanding of the customers' needs.

Finally, the beam pattern algorithms were implemented one at a time and unit tested. Features such as mode of operation, and time delay were implemented last. The final stage consists of verifying the result with the theory and generating a typical degraded array beam pattern to estimate the effects of individual element degradation.

A small feasibility study was done on the theory and on the software program itself. First, discussion between Dr L.J. Ziomek and the author established that it was possible to produce a rapid algorithm to generate beam patterns. It was established at that time that an FFT computer algorithm (or hardware board) could be used to do so.

Initially, it was estimated that about 4 or 5 "student-months" would be required to produce the software program. The stages were to develop the algorithms for a planar array and a cylindrical array. Then a spherical array could be added to simulate the nose. A conformal array's algorithm could be made from the simple geometry (planar, cylindrical, spherical) or from basic theory. In this case, the algorithm was derived from basic theory instead.

During the feasibility study, the size of the program (delivered line of source code) and time requirement for developing the program was defined, using sample programs available from Hewlett-Packard Corporation, LT R. Self's basic graphics packages, and the author's experience. This is shown in table II (no memory limitation has been considered).

A list of priorities was also established:

TABLE II

INITIAL DEFINITION OF SOFTWARE PRODUCT

Routine	Time Required	Size (DLSC)
1. Learning language	3 weeks	----
2. 3 Graphics routines	4 weeks	1500 lines
3. Database definition and Initialization	2 weeks	500 lines
4. User Menu		
- Input (& Validation)	3 weeks	500 lines
- Graph output	3 weeks	400 lines
- Individual data entry	2 weeks	300 lines
5. Table of Input	1 week	100 lines
6. Table of Output	1 week	100 lines
7. Array position definition (and Look up tables)	2 weeks	300 lines
8. Time Delay	2 weeks	300 lines
9. Array Definition	2 weeks	200 lines
10. Beam Pattern Calculation	4 weeks	600 lines
11. Window - Mode of Operation	1 week	100 lines
12. Documentation in code	4 weeks	(included in previous routines)
	34 weeks	4.9 Klines
	about 8 student-months	

where DLSC = Delivered Line of Source Code

1. Generate theoretical and then degraded beam pattern curves with documentation for a given MRA, frequency, speed of sound, and mode of operation;
2. Design an input MENU;
3. Develop three graphics formats, and generate the tabular output;
4. Input the actual sensitivities using a computer file;
5. Develop a MENU for individual input of complex sensitivities for testing purposes.

The following features were felt to be desirable but only if time allowed.

1. For a small number of MRA, produce a graph showing degradation (angles and ranges) of the main lobes only. This should allow a quick evaluation of the general array status;
2. Input "Look-up" Tables of other submarine's arrays, i.e., implement many classes of submarines that are similar shape to the one presently being implemented;

3. Implement consideration for the diffraction of plane waves around the submarine;
4. Produce a statistical package that keeps track of actual degradation between overhaul refits of the array;
5. Simulate aging of newly installed elements;
6. Enter real beam pattern of individual elements (when available) using the product theorem;
7. Simulate more features of the Beamformer; and
8. Use shading to improve detection, and propose new modifications (enhancements) which can be made on the array.

Basically, the main objectives of the thesis will be implemented plus some user-friendly features.

D. ANALYSIS - PLANNING THE DEVELOPMENT PROCESS

Since this is a one programmer program, there was no detailed organization of the structure of the project, nor any detailed life-cycle model. As indicated in the first section of this chapter, a combination of Boehm's and Fairley's life-cycle models were used. Excluding maintenance, which is usually the last (and longest) phase of the life cycle management, every phase mentioned in section one is performed sequentially.

After the analysis phase, the product design is verified against the software plans and requirements and different aspects of the feasibility study to determine if any changes are required. The detailed design is then verified against the general product design. During the implementation phase, each section of the design is encoded individually, then unit tested, debugged, and again tested and verified against the design. After its implementation in the main program, more testing and validation is done. During system testing, the integration of the whole program is verified against the code done previously and then validated against

the details of the analysis phase, i.e., program requirements, the goals, the planning, and, moreover, against the theory being used.

If a major mismatch occurs during the unit testing for the acceptance test, a lot of work can be involved since many phases have to be updated or even repeated. The maintenance phase, the final stage, involves more revalidation since the product would be operational.

The development process is complex and detailed. Therefore, only the essential points are described. Table II provides the estimated time required.

In March 1985, the initial definition of the requirements was done. By April, the thesis proposal was submitted, the study of the language to be used was completed, and the initial hardware requirement was identified. By May, a simple prototype had been developed using the HP BASIC structured language. The prototype was, as mentioned before, a line array of complex weighted point source elements (odd number), where beam steering and amplitude shading were performed. No MENU was available then, and only the rectangular plot of the normalized magnitude was produced. An FFT algorithm was then used to accelerate the processing time.

E. SOFTWARE REQUIREMENTS

This section specifies the external behavior of a system, without implying a particular implementation. Ideally, this is about the "What" to expect not the "How" to do it. In fact there is some discussion of how this was to be done, such as some of the algorithms that must be used, the language used, etc. Fairley [Ref. 11: p. 89] provides a very detailed list of software requirements specifications that should be provided by the user; however, this is not practical in this case. Therefore, only a few specific aspects of the requirements will be addressed, since the previous section already highlighted the "what" to expect.

1. Computer Characteristics and Configuration

The computer system used is Hewlett-Packard (HP) 9000 series 200 Personal Technical Computer. This micro-computer series is used extensively by many universities and laboratories. The series has some compact models; however, none of these micro-computers are considered "portable computers". In order to do graphics, a graphics CRT must be used. A strict minimum of 1.5 Mbytes of random access memory (RAM) is required. A larger size is recommended, in the range of 2.0 Mbytes. The models in this series have a M68000 (16 bit) microprocessor and a clock working at a rate of at least 8 MHz.

The keyboard is standard with the addition of a rotary knob and 8 special function keys (all used by the MENUs).

At least one HP9121D flexible double density single sided 5-1/4 inch floppy disk drive is required. The configuration uses for development used two disk drives. The capacity of a disk is 270 Kbytes. The use of a Winchester disc (hard disc) would accelerate the loading process since a minimum of 4.6 Mbytes is then available on one hard disc.

The configuration used for software development was a HP model 236 computer. The set up requires an additional hardware electronic board the Infotek FP210, 10 MHz Floating Point Coprocessor, that greatly enhances the performance of the system. For example, when using the HP BASIC language, a sine and a cosine operation is done 7.9 times faster, an exponential 7.5 times faster, and a division 3.8 times faster. It is a user transparent coprocessor which requires no modification to existing programs and operates both HP BASIC and PASCAL. It is also almost twice as fast as the Hewlett-Packard Math Card. However, the Infotek's BASIC compiler may be used with the FP210 FPC. Any compiled program is considerably faster than an interpreted one. A

factor of two to 22 and even 50 can be observed in the speed of execution when using the compiler. The compiler is also easy to use. The cost of the Infotek's Floating Point Coprocessor and BASIC compiler is modest compared to the cost of the hardware.

For hard copy, a printer should be available. Since graphs will be generated, the use of dot-matrix, ink-jet, laser-printer or thermal printer is recommended since an impact or a "Daisy-wheel" printer cannot reproduce high quality graphs. The HP2225A Thinkjet is suggested for its speed and performance.

For plotting, i.e., for a better quality graph output, any plotter compatible with HP and having at least two pens can be used. The HP7470A is the most economical. The port used for the plotter is address 705 (see next sub-section).

2. Hardware Interfaces

All the microcomputers in the HP 9000 series 200 come with a Hewlett-Packard interface bus (HP-IB), Hewlett-Packard's implementation of the IEE488 interface bus. Normally, the peripherals are connected to the bus, as in this project.

The interface HP-IB can support a maximum of 31 devices representing 31 addresses (700 to 731) and 15 standard device loads. The interface itself represents one address and one load. Each device connected to it may represent one or more addresses and one or more device loads. For example, if an ink-jet and a thermal printer are both turned on with the same address (say 701), any data sent to address 701 will be printed by both printers.

The following addresses will be used in this program:

1. 701 - Printer;
2. 705 - Plotter; and

3. 1 - CRT. (Note: not 700 + 1)

The reader should refer to the appropriate technical documentation provided by HP for the detailed characteristics of these devices.

3. Software Language and Operating Systems

The language used for the implementation is the Hewlett-Packard Technical BASIC language version 3.0. It is a structured language (third generation) like PASCAL. Indeed, HP BASIC is an amalgam of PASCAL, C, and FORTRAN, but with a very limited set of data types. HP BASIC possesses an extensive number of statements, commands, functions, and operators that are all treated as keywords.

HP BASIC is simple to learn, use, and maintain. It has most of the characteristics of a structured language such as context definition, data abstraction, higher control structure such as if-then-else, for-do loop, repeat-until, while-do, etc., and the statements can be indented for a better visualisation of the levels [Refs. 13,14].

Only four data types are available; real, integer, string, and integer or real number arrays. This limits the capacity of the language but it is still adequate for our processing. Meaningful variable names are available and may be used to clarify meaning or usage. Up to fifteen alphanumeric characters can be used for each variable. With HP BASIC, data declaration can be done at any point in the program. This can seriously weaken control of the data-base, especially during the maintenance phase. Therefore, all data declarations should be done in one section: the data-base definition subroutine. Exceptions such as local variables for subprograms and functions are normal, and will be accepted. Furthermore, HP BASIC automatically creates a variable during an assignment if this variable was not already defined. This commonly occurs when a variable is misspelled. Although this feature is not too hazardous in a

small program, it can greatly threaten the accuracy of execution of a large program. A cross-reference must be produced and compared with the data-base in order to identify any misspelled, doubly-defined, or inadequate data or variables. A separate module (subroutine) must be used for data initialization and data definition.

The HP BASIC language comes with its own operating system called the HP Language System. Additional systems exist, such as the Shared Resource Management (SRM) operating system and the HP-UX operating system. The standard HP BASIC language system is best suited for use instead of HP-UX or SRM.

Moreover, for graphics generation, a language extension is required. The HP Graphics Language (HPGL) has been designed for that purpose. The language extensions of the BASIC 3.0 used for mathematical operations, input/output, etc. mentioned in table III may be used to further enhance the system.

TABLE III

BASIC 3.0 LANGUAGE EXTENSIONS OF THE PROJECT

NAMES		SIZES (of RAM)
RAM-Based Basic 3.0		250 K
with		
GRAPH	- Basic graphics	41 K
GRAPHX	- Graphics Extension	23 K
IO	- Extended I/O	11 K
TRANS	- Transfer statement	34 K
MAT	- Matrix operations	21 K
PDEV	- Program development	14 K
XREF	- Cross Reference statement	6 K
KBD	- Keyboard extension	12 K
CLOCK	- Real-time clock	4 K
MS	- Extended mass storage	10 K
ERR	- English error message	7 K
HPIB	- HP-IB	12 K
Serial	- RS-232	6 K
CRTB	- CRT memory plane	18 K
		469 Kbytes

4. Timing Constraints

The software program must produce its output rapidly because it is intended for operational use. The actual calculation of both theoretical and degraded array case must be done in less than three minutes of processing time. The execution time of this calculation should be shown on the screen. Since MENU execution speeds are limited by the speed of the user, error checking on data input should be transparent to the user with a maximum waiting period of two seconds per entry.

Graphics generation should be limited to about ten seconds per entry.

The sequencing of the program modules should be such that the user is not exposed to long periods of dead time, with the exception of the beam pattern calculation where the message "computing" should be displayed. A user should not wait more than 6 to 10 seconds (dead time) without an on-screen notice of which function is being performed. This does not apply when the program is waiting for user input or is in the "PAUSE" mode (which is indicated to the user).

The overwhelming concern is the speed of the I/O units. Printing a table of the magnitude of the beam pattern for every degree may require a few minutes. The same applies for a graphs dumped on a printer or a plotter. However, no control of this time limitation can be accomplished.

5. Accuracy Constraints

Double precision should be used throughout the program in order to obtain accurate data, since the answer of the DFT is sensitive to small variations in the phases.

A value of the beam pattern for each degree is needed for a sharp definition of the graph. Finally, no rounding off should be done except for conversion to integers when required.

6. Structured Design, Structured Programming, and Modularity

Since HP BASIC 3.0 is a structured language, a structured design and a structure programming approach must be used. This contrasts with lower level languages, such as FORTRAN IV or WATFIV, where "spaghetti coding style" is common, even if it does not need to be. This style is basically full of branches using GOTOs, with multiple entry and exit points, and is extremely difficult to maintain due to the lack of clarity and structure. This is true even if the program performs well.

Fairley [Ref. 11: p. 181] indicates that two design techniques are recommended for scientific applications: structured design and stepwise refinement. A structural design approach will be used in this thesis since it is a well documented and tested technique. Myers [Ref. 15: p. 69] developed a technique called composite/structured design or simply composite design which is a top-down technique for external and architectural design. It is this technique that will be used. Constantine and Yourdon [Ref. 16] suggests an alternative procedure similar to this one that is also adequate.

Composite design is concerned with the structure attributes of a program, in terms of module, data, and task structure. Composite design has no direct relationship with coding structures (detailed design) or memory structures. It has no direct relationship with the performance of the system, nor its actual function from an external point of view.

The basic approach in composite/structured design is that the structure of the program should resemble the structure of the problem (see software requirements). To accomplish this, composite design involves an analysis of the problem structure, the flow of data through the problem, and the transformation that occur on that data. Data flow

diagrams are then developed, and then refined to represent the system. The system is then decomposed into sub-systems or modules using heuristics such as module coupling [Ref. 15: Chap. 4] and module cohesion [Ref. 15: Chap. 3]. A well designed system exhibits a low degree of coupling between the final modules and a high degree of cohesion among elements in each module. This helps minimize the number and complexity of the interconnections.

After the detailed data flow diagrams of the system are made, they are converted into structured charts (see next paragrahs). Some of the data structures can be specified at that point. The next step is usually to refine the structure charts, next to convert them into pseudocode and then into program code. These last steps are parts of the detailed design and the implementation. The software design section provides the definition of these terms along with the resulting diagram and charts for the program.

After the design, the program is coded, and structured programming should be used during the implementation of the program.

Structured programming is a method of coding a program using a limited number of control structures, such as if-then-else, do-while, repeat-until, case statements, and sequencing. A program written this way has the property of having virtually no statement labels to be used by GOTOs. If these rules are combined with an additional rule that states that each segment (or module) has exactly one entry point and one exit point, mathematical proof of the program's correctness as well as its testing is greatly simplified.

Structured programming significantly improves the clarity, readability, and maintainability of a module because the structured code is literally read from top to bottom. There is no jumping around on the program listing,

which is, as mentioned earlier, a typical fault of programs containing GOTO statements. Structured programming also lends itself to the use of indentation rules [Ref. 15: p. 110].

In structured programming, the code is divided into modules. These modules preferably represent the functional units of the program. This process of modularization helps to impose hierarchical ordering on function usage, to implement data abstractions, and to develop an independently useful system [Ref. 11: p. 147]. Examples of modules include individual procedures, subroutines, subprograms, and functions; functional groups of related procedures; data abstraction groups; utility groups; and concurrent processing if any.

It is conventional practice to define a module and its sub-modules as corresponding to the processing steps in the execution sequence. For example, a module should be used for each menu and graph type, the data-base definition, the computation of a beam pattern, etc. This illustrates the chief advantage of modularity, namely the ease with which the program may be updated or revised. Thus, any module can be changed at a later time, during the maintenance phase, without affecting the rest of the program. Of course, any changes to the common data-base, for example, must be reflected in all modules using it.

An additional simplification is to break modules into segments. Each segment is a piece of code with about 50 or fewer statements and as much as possible has one entry point and one exit point. The main exception is that softkeys for the menu are defined with GOTOs and multiple entry points. The softkeys are ten special keys available on the HP9000 series 200.

Finally, every module should be documented in code. Immediately following the entry point of a module, i.e.,

after the function or sub-program name, or subroutine label, a standard format proloque should be available. The following list describes the information contained in the proloque:

1. Module name: describing the function performed by the module;
2. Date: date of implementation (version 1.0);
3. Programmer name: initial person or team which implemented the module;
4. Purpose: a one sentence description of the function performed by the module. If necessary, this is followed by an expanded description. Note that only the module's function, not its internal logic or operation, should be described here.
5. Input: name of all the input variables used by the module (or passed to it);
6. Output: name of all output data and variables from the modules. It includes tables and graphs.
7. Side effects: (and external effects) this refers to any action that manifests itself outside of the program, such as a message on the screen, a printout, erasing of an external file, etc. Also any important features, like resetting of flags and control variables, etc., should be mentionned; and
8. Remarks: any clarifying remarks which may help the understanding of the modules, such as theory used, units used, etc.

Most requirements definitions contain a simplified data flow diagram. This will not be the case here in this thesis. All diagrams relevant to the design will be in the design specification section.

7. Menu

This software program requires three menus. One menu permits the user to enter all the initial parameters and to request the type of output (tables or graphs). A second menu allows individual changes of stave condition (complex weights changes)to be entered, and the third menu allows the choice of the medium on which the requested output (if any) will be reproduced (printer, plotter, etc).

All menus should be self-explanatory, i.e., enough information must be provided for the user to make an intelligent choice. In order to prevent run-time errors due to improper data entry, such as "thirty" instead of "30", data validation must be performed on all inputs. A complete and thorough validation of all inputs will tend to insure the program's robustness.

In case of an validation error, enough information should be provided to the user about exactly what errors occurred so the errors can be understood. For example, if a user enters alphabetic data where a number is required, a message stating that a number must be entered should be displayed. Then the user should be allowed to resume his entry. This legality validation (syntax) is an essential part of the project and makes it more viable operationally.

Validation of the input must check that it is within the bounds set by either the problem specification or physical reality. If a negative number is entered where a positive datum is needed, then the menu should inform the user about this error. As examples, the program should prevent the user from entering a negative frequency or a speed of sound of 15,000 m/s. This is called plausibility validation (semantics) and must be done throughout the program.

If any improper input slips through the input program checks for legality and plausibility, there is one final checkpoint--the user. By echo printing the input

data, the program allows the user to locate incorrect data before proceeding further. Certain functions are not echoed-printed, e.g., the request for output on a printer.

The following paragraphs will list the minimum requirements for all three menus. The entry menu should have the following features:

1. The choices of parameters along with present defaults are displayed as a whole on the screen;
2. A choice for tabular outputs of the aperture function and/or the beam pattern data as function of horizontal angles. This is in the form of a yes/no display;
3. A choice of any or all graphics options, i.e., normalize magnitude of beam pattern, magnitude of beam pattern in dB, and polar plot of beam pattern in dBs. Again a yes/no display is preferred;
4. Positive real frequency in Hz with default of 1000 Hz;
5. Speed of sound in the media in m/sec within a range of 1400 to 1600 m/s with a default of 1500 m/sec;
6. Choice of SUM or DIFF as the mode of operation. The user should only have to use one key to make one choice, i.e., no long typing should be required. The default is to be the SUM mode;
7. Finally, a "START" key is necessary to proceed after all parameters have been entered and are satisfactory to the user.

The second menu allows the user to change the complex weights of any elements of the defined sub-array at a given MRA. Many smaller menus can be used. The following features are required:

1. A choice between changes or no changes at all of the complex weights;

2. A stave number can be selected, then one or all its hydrophones can be chosen for changes;
3. Provide choice between dead or disconnected staves, padded staves, or user defined complex weights, with a phase between 0 to 360 degrees, and a magnitude between 0 to -40 dB;
4. Provide the current and the new complex weights of the stave being changed (except for disconnected staves); and
5. A special key to get out of the menu must be available at all times;

Again enough information must be provided to keep the user informed of what is happening at all times.

The last menu deals with the graphical output medium. For each graph requested, the user should be given a choice of different media, such as the printer, plotter, etc., for output. The minimum options requested are:

1. CRT (screen);
2. Printer; and
3. Plotter.

Additional features such as size of graph (normal or expanded (4 times the normal size)), and the type of peripherals (such as a thermal or inkjet printer) could be implemented.

Again, for this last menu, the user should not have to type explicitly any medium, i.e., a key menu should be used to minimize input error.

8. Graphics

Three graph types must be implemented in the first version of the software program. If the operator fails to request any graphical output, a message should be displayed following the beam pattern calculation, and again following the output of the tables (if applicable), informing him that "as requested no graphs will be generated".

The program must produce graphs that have a title, a legend, and the MRA in degrees clearly indicated. At the bottom of the graph, a line of printing should indicate the mode, the frequency in Hz, the speed of sound in m/sec, and the date (optional). It should also be labeled properly, e.g., 1000 Hz.

The problem of labeling and numbering, in appropriate units, the abscissa and ordinate in rectangular plots should be addressed. The units used must be indicated next to the label of the axis. For polar plots, the angles shown on the graph should be displayed around the graph and the unit of the radius should be shown on one of the axes (here the units are dBs).

The horizontal angle in degrees is based on the definition shown in Figures 2.1 and 3.1 . Therefore, it will vary from -180 to +180 degrees. In this scheme, 0 degrees is straight ahead toward the nose of the submarine, where 90 degrees is starboard (right of the submarine), and -90 degrees is port side (left side), and 180 degrees is toward the rear.

The three types of graphs to be implemented are:

1. Normalized magnitude (0 to 1.0) of the far-field beam pattern as function of the horizontal angle;
2. Normalized magnitude in dB (0 dB and less) of the far-field beam pattern as function of the horizontal angle; and
3. Polar plots of the normalized magnitude in dB of the far-field beam pattern.

The graph of the normalized magnitude of the beam pattern shall vary from 0 to 1.0, where the highest maximum is set to 1.0.

The graph of the normalized magnitude of the beam pattern in dB shall vary from 0 to -40 dB, where 0 dB is the maximum value of the highest peak of the beam pattern curve.

The polar plot graph is simply the rectangular normalized magnitude in dB in polar representation.

All plots shall show both theoretical and degraded array far-field beam patterns. A different line type for plotting must be used to identify the curves and should be defined in the legend.

9. Changes and Modifications

During software maintenance, changes and modifications will be implemented from time to time. The software code (program) should reflect these changes. Such changes should be thoroughly annotated in the proloque of each function, sub-program, and sub-routine affetced. The notation should include a new version number, eg., V1.2, the name of the programmer making the change, the date, and a summary of the changes made, along with the reasons and authorization for the change. The actual change in the code can be identified with a comment containing the version number and the initial of the programmer, e.g., V1.2 SF. This commonly used technique will be applied to already existing routines extracted from other software programs and adapted for this software project.

10. Important Assumptions

The program uses the theory formulated in chapter two. Equations 2.97 to 2.99, which describe the calculation of the beam pattern, form the basis of the program. The assumptions of chapter 2 are also applicable here. The coordinate system of Figures 2.1 and 3.1 will be used for all versions of the software. Furthermore, no diffraction considerations are taken into account. The speed of sound is constant over the span of the entire array. Up to four sub-arrays (portions of the main conformal array) can be defined at one time for a given MRA. With the present implemented conformal array, no more than three sub-arrays can be defined for a given MRA.

To simplify future upgrades, the program should incorporate a feature that allows important constants such as maximum number of staves, minimum dB level, maximum horizontal angle to be implemented as parameters. Currently, the maximum number of staves is set at 52, the minimum dB level is -40 dB, and the maximum absolute value of the horizontal angle is 180 degrees. The type of lines used in the graphs is also a variable.

F. SOFTWARE DESIGN

Design is the process of defining the software architecture, components, modules, interfaces, test approach and data for a software system to satisfy specified requirements. In software design, three types of activities can be distinguished: external design, architectural design, and detailed design. These last two are referred to as internal design.

External design of software involves conceiving, planning, and specifying the externally observable characteristics of a software project [Ref. 11: p. 137]. Architectural design is concerned with refining the conceptual view of the system, identifying internal processing functions, decomposing high-level functions (or modules) into subfunctions (or submodules), defining internal data streams and data stores, and establishing the relationships and interconnection among functions, data streams, and data stores. It describes the structure of the system. Detailed design deals with the specification of the algorithms that implement the functions, concrete data structures, actual interconnection among functions and data structures, and the assembly scheme of the software project [Ref. 11: p. 138]. Therefore, it describes the control flow, data representation, and other algorithm details within the modules.

External design and architectural design will be described in this chapter, while the detailed design is discussed in Appendix A.

The composite/structured design explained in the subsection called structured design, structured programming, and modularity of the software requirements, is used for this program. It consists of systematically developing the data flow diagrams of the program, which are then converted into structured charts,to pseudocode, and then to the final software code.

Before proceeding further, the words, data flow diagram, structure chart, pseudocode, and structured flowchart will be explained.

Data flow diagrams ("bubble charts") are directed graphs in which the nodes specify processing activities (modules) and the arcs specify data items transmitted between processing mode [Ref. 11: p. 152]. They can be used at any desired level of abstraction. Unlike the traditional flowcharts, data flow diagrams do not indicate decision logic or conditions under which various processing nodes in the diagram might be activated, i.e., there is no control flow.

Structured charts express the hierarchical structure, parameters, and interconnections in a system. While a data flow diagram gives the logical picture or paths of the system, the structured chart gives the physical structure of the system. A structured chart has no decision boxes, and the sequential ordering of tasks inherent in a flowchart is suppressed. As it is refined, the chart can be augmented with more modules. During this step, the parameter attributes are abstract; they are refined into concrete representations during the detailed design.

Pseudocode describes the module characteristics using short, concise, English language phrases that are structured by keywords such as if-then-else, while-do, repeat-until, for-to, begin-end. Keywords and indentation describe the flow of control, while the English phrases describe processing action [Ref. 11: p. 158]. Furthermore, each

English phrase can be expanded into more detailed pseudocode until the design specification reaches the level of detail of the implementation language. Pseudocode is extensively used in the detailed design phase.

The flowchart is the traditional means for specifying and documenting algorithmic details in a software system. Many different symbols are used, along with the representation of the control flow. When using a structured language, structured flowcharts can be used. Structured flowcharts (a special type of flowchart) are restricted to composition of certain basic forms. This makes the resulting flowchart the graphical equivalent of a structured pseudocode description. The basic forms are characterized by single entry into and single exit from the form. Structured flowchart allows hierarchical nesting of structured flowchart constructs to document a design in top-down fashion, starting with the top level structure and proceeding through detailed design. A structured flowchart shows the flow of control mechanisms, i.e., decision mechanisms and sequencing of the control flow.

The composite design approach will be applied to the project in the following sections.

1. Design : A Simplified Data Flow Diagram

The approach is based on the structure of the program, i.e., what the program has to do without regard for procedure, time, sequence, or which event has to happen first.

Myers [Ref. 15: p. 70-71] identifies the initial steps to follow as:

1. Identify the problem structure;
2. Identify the principal input and output data streams (but not all the data);
3. Identify the points in the problem where these data streams begin and end;

4. Break the problem into a set of subordinate modules; and
5. Repeat the process iteratively, viewing each module from step 4 as a new subproblem.

The far-field beam pattern simulation program is a transformed-centered program where the parameters provided by the user are entered, then the required processing data are computed and extracted from the data base (tables, or files, etc.), they are then "transformed" into an aperture function which is "transformed" into a beam pattern function. Finally, the results are processed for output.

The principal inputs to the program by the user are:

1. Frequency f (Hz);
2. Speed of sound c (m/sec);
3. MRA (Maximum Response Angle - Degrees);
4. Mode of operation (SUM or DIFF);
5. Output required (tables, graphs, and on what medium); and
6. Stave numbers with their changes or new status (new relative complex sensitivities, padded, or disconnected).

The principal outputs are:

1. The complex aperture function of the conformal array (both theoretical and actual (degraded) cases); and
2. Beam pattern functions for both theoretical and actual cases in a tabular form and in a graphical form.

The major data groups, or tables, required for a given configuration but not entered by the user are:

1. The definition of the staves used for a given MRA, i.e., the sub-arrays;
2. The spatial locations of the hydrophones of the conformal array;

3. The complex weights or the relative complex sensitivities of all the hydrophones as function of frequency; and
4. The time delays applied to the sub-arrays defined by a MRA (that are a function of the speed of sound).

Disregarding the database definition and initialization, and similar standard modules, three basic modules can be identified:

1. One which chooses the beam pattern parameters (the menus);
2. One which extracts all the necessary array data, i.e., creates the aperture function of the conformal array; and
3. One which computes the far-field beam pattern of both the theoretical and the actual array.

Figure 3.8 shows the resulting simplified data flow diagram of the program. Again, the circles or nodes represent the modules or processing elements that transform data; the arcs represent the data links between nodes. The structure between two horizontal bars with a name, represents the major data groups used in the program where data are extracted or stored.

The arcs on the left are the principal inputs, those on the right represent the principal outputs. The user has no direct access to the major data groups, the menus will do the required selections.

2. Design : Main Data Flow Diagram

The next step is to refine the simplified data flow diagram of the system. Fist, the leftmost module "Choose beam pattern parameters" can be divided into three menus as specified in the design requirements. The first selects the initial beam pattern parameters, i.e., frequency, speed, MRA, mode of operation, the output required (tables and graphs), along with a flag to request execution of the

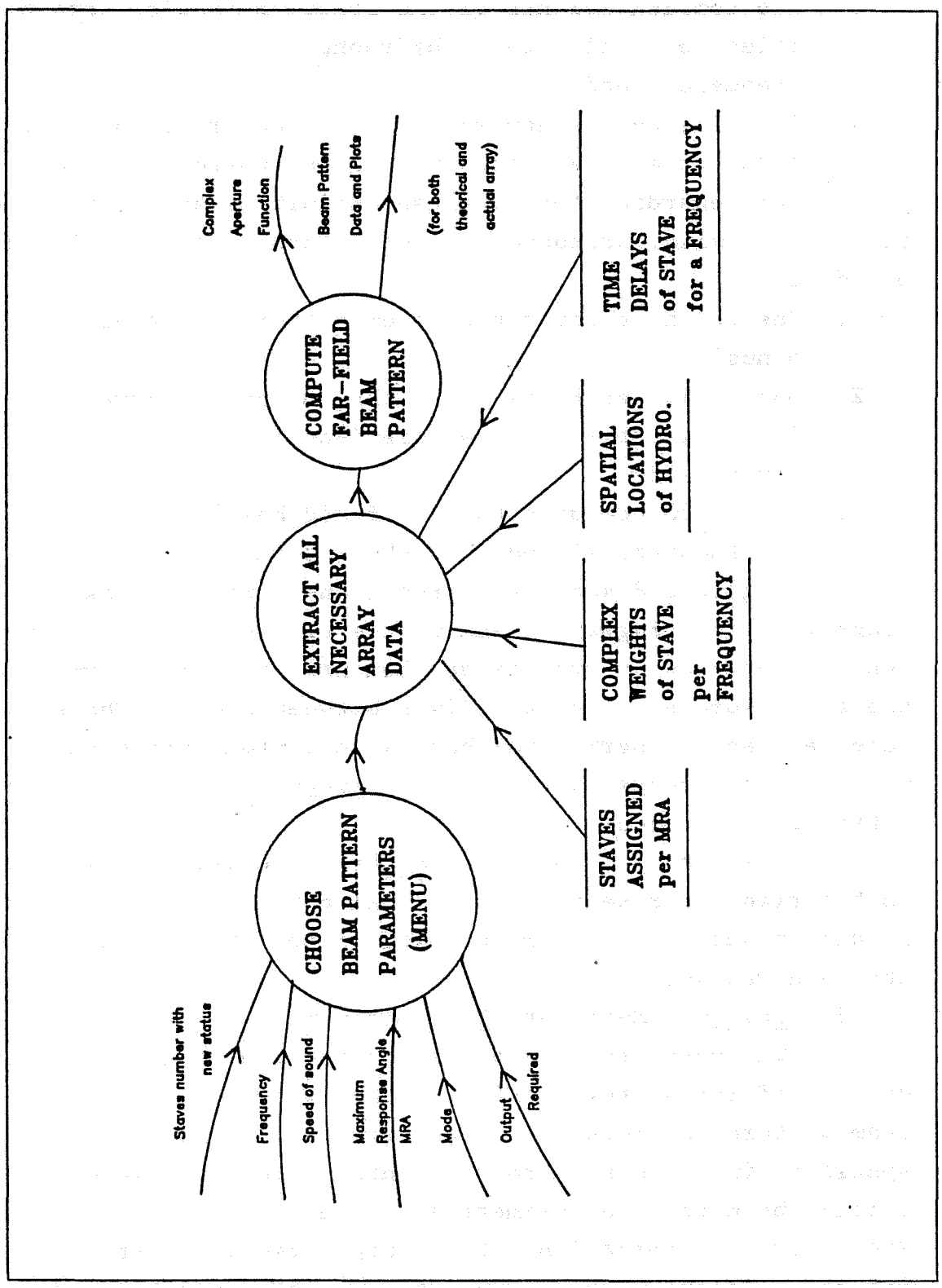

Figure 3.8 Simplified Data Flow Diagram of the System.

program. The second menu allows the operator to select a stave or hydrophones and to modify its status. The new status can be a new value of complex sensitivity, or padding or disconnection of a stave or hydrophone. The phrase "new status" will be used. This second module is called "Modify individual staves". Finally, a third menu, "Choose graph output medium" allows the user to decide which medium the plots use (printer, plotter, etc.).

The middle module of Figure 3.8, called "Extract all necessary array data", basically retrieves all necessary array data to create the aperture functions of both the theoretical and the actual cases. This is done by selecting the correct staves for a given MRA, specifying the location of all the elements along with their actual and theoretical relative complex sensitivities, applying the correct time delays to each staves, and finally applying the selected aperture window (the mode of operation) on the sub-arrays. Moreover, the new status of any selected staves must be reflected in the actual array aperture function.

The previous module of Figure 3.8 can then be decomposed into three more modules as shown in Figure 3.9 . The "Configure array" uses the frequency, the MRA, and any "new status" to define the staves used, the complex sensitivities or weights of the elements as a function of frequency, and the locations of the hydrophones from the data store. The resulting preliminary aperture function, along with the MRA, and the number of elements N (per sub-arrays) is used by another module, "Apply time delays", which extracts the time delays applied at a given MRA from the data store and builds a new aperture function. Finally, this aperture function is filtered by the module "Apply window on array" which extracts the parameters of the selectd window (mode) from a data store. The result is the final aperture function for both the theoretical and the actual array.

The module "Compute normalized beam pattern" uses the defined aperture function, the MRA data, and N to generate the beam patterns along with the horizontal angles. These beam patterns can be outputed directly or sent to the module "Generate plots of beam pattern" as chosen by the operator on the selected medium. Three graph types are available as specified by the requirements which is illustrated by the three exiting arrows on the figure.

Figure 3.9 shows the resulting main data flow diagrams of the system.

3. Design : Main modules

Before showing the resulting structured charts which illustrate the actual structure of the system, more data flow diagrams are provided to illustrate the main modules of the program.

The first is the module, "Compute the normalized beam pattern". Figure 3.10 shows the details of the module. Again, the entering arrows indicate the input, the exiting ones indicate the output. The data flow diagram for the module, "Configure array", is described in Figure 3.11 .

The menu "Choose beam pattern parameters" (User menu) is simply an interface, allowing and validating user entry. Figure 3.12 shows the details of this menu.

The menu "Choose graph output medium" allows the user to choose the medium for each graph type requested. Figure 3.13 shows the corresponding data flow diagram.

The last menu is more elaborate. Figure 3.14 shows the detailed data flow diagram of the menu called "Modify individual staves".

The modules, "Apply time delays" and "Apply window on array" are discussed in Appendix A. The last module of Figure 3.9 which is "Generate plots of beam pattern" is simply a sequence of outputs on the screen. It is easier to describe it with a structure flowchart or a structured

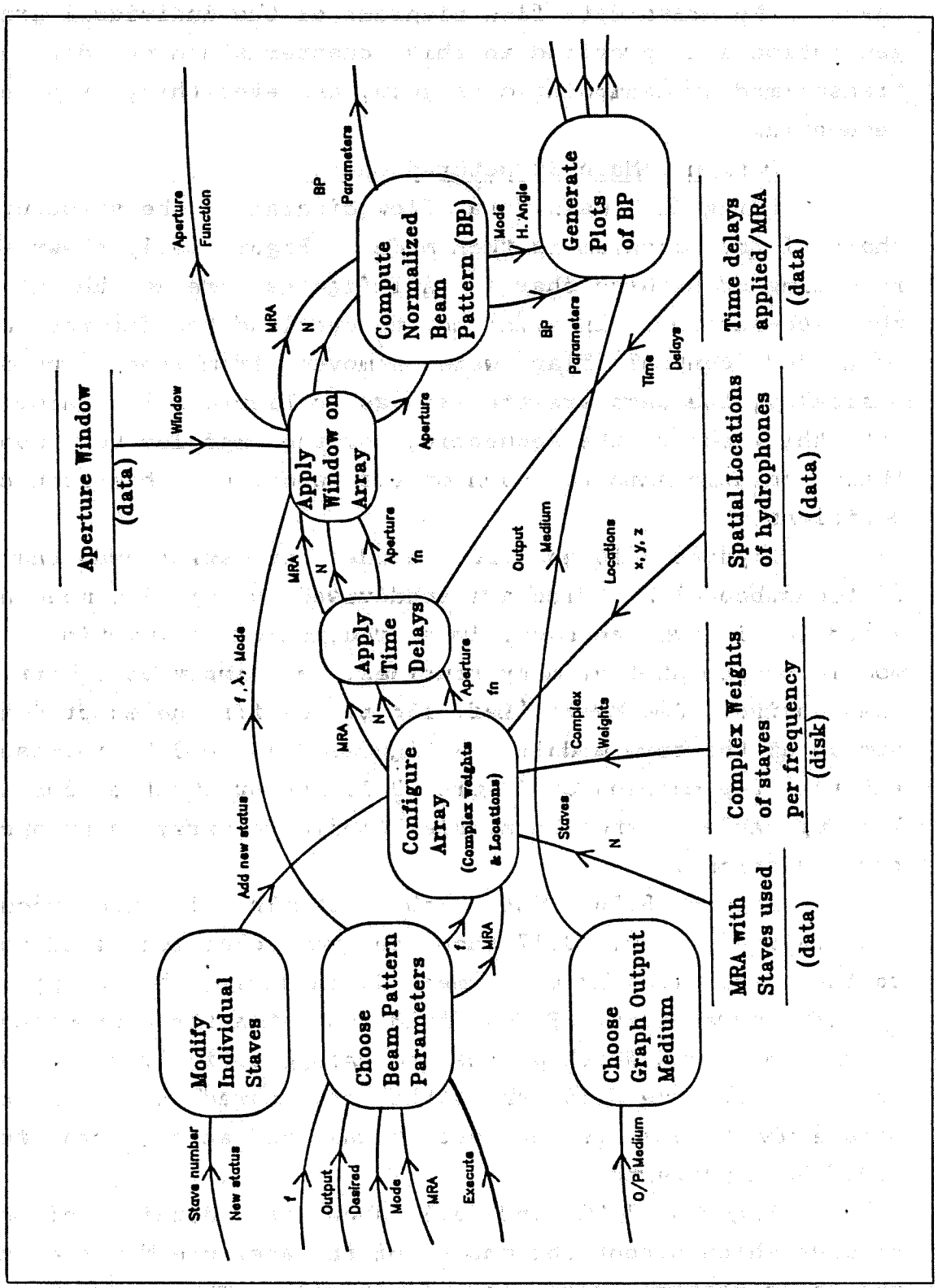

Figure 3.9 Main Data Flow Diagram of the System.

chart. No graph data flow diagrams of the individual graph generation are provided in this chapter since no data are transformed or manipulated as such, and everything is purely sequential.

4. Design : Main Structured Charts

Using the main data flow diagram, the structured chart of the program is then made. Figure 3.15 shows the resulting structured charts. All figures are at the end of this sub-section. In order not to overload the figures, all data and control flag were removed from the figures. Basically, the data are the same as in Figure 3.9 Again, do not think about the sequencing of the modules but about their relationship to each others and their hierarchical positions.

Figures 3.16 to 3.22 provides more structured charts of the submodules which are subdivised in smaller modules. A module is not necessarily a routine or a function. A module can consist of many routines, or simply be a part of one routine. See Myers [Ref. 15: p. 11] for the exact definition of the term module. Figures 3.16 to 3.22 represent all the top modules of Figure 3.15 except the two for the look-up tables, and the module "Configure array as an aperture function".

Figure 3.16 shows the details of the module "Initialize". Figure 3.17 shows the ten selections available to the user with the user menu which allows the change of specific parameters. Figure 3.18 shows the structured chart of the next menu which allows the change of individual stave status. Figure 3.19 shows the structured chart of the module which computes the theoretical and actual array far-field beam patterns.

Figures 3.20 and 3.21 show the details of the modules which output the tables of the aperture function and the beam pattern data (or function). The modules are executed even if no requests for a table were made.

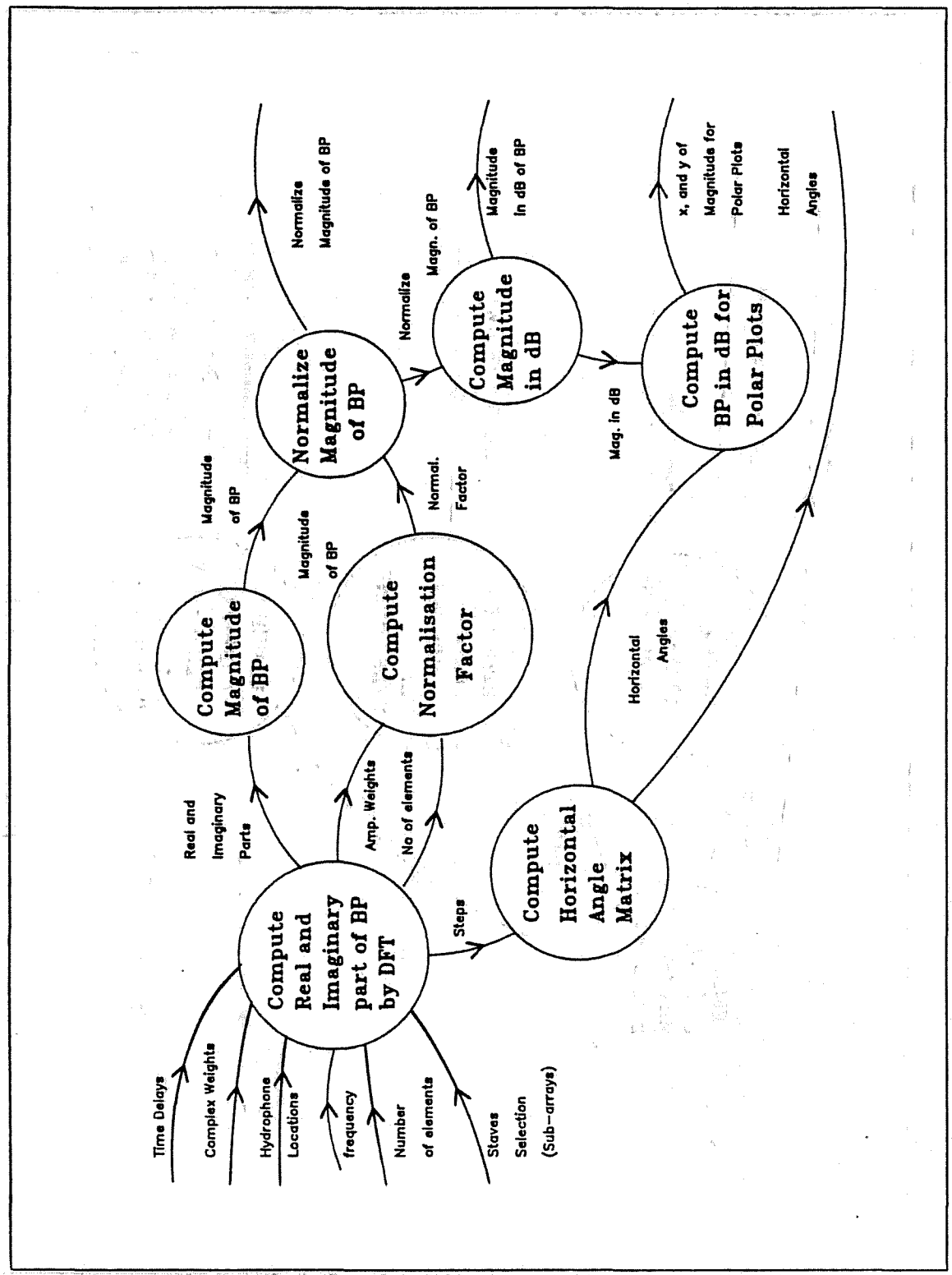

Figure 3.10 Data Flow Diagram of "Compute Beam Pattern".

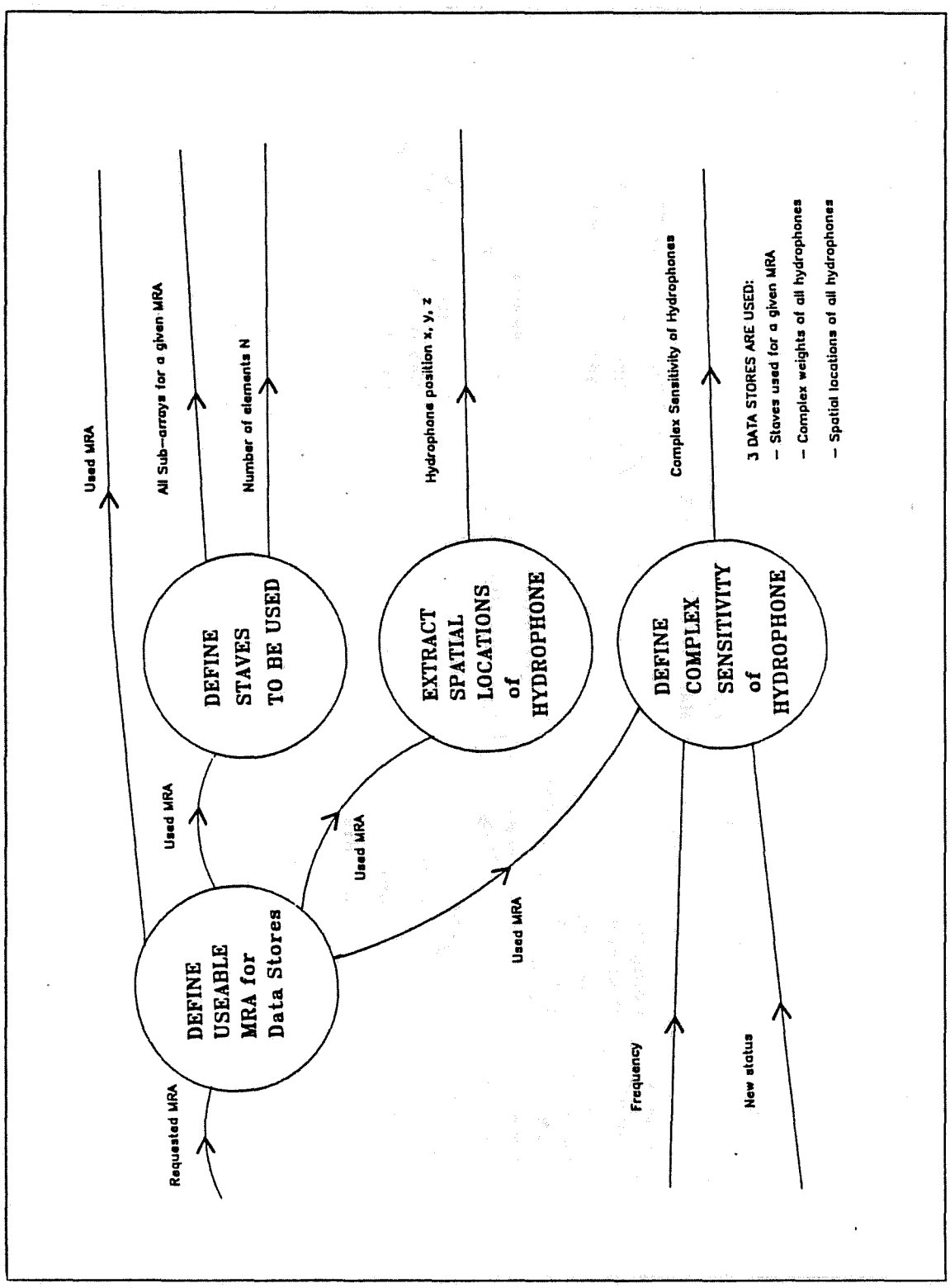

Figure 3.11 Data Flow Diagram of "Configure Array".

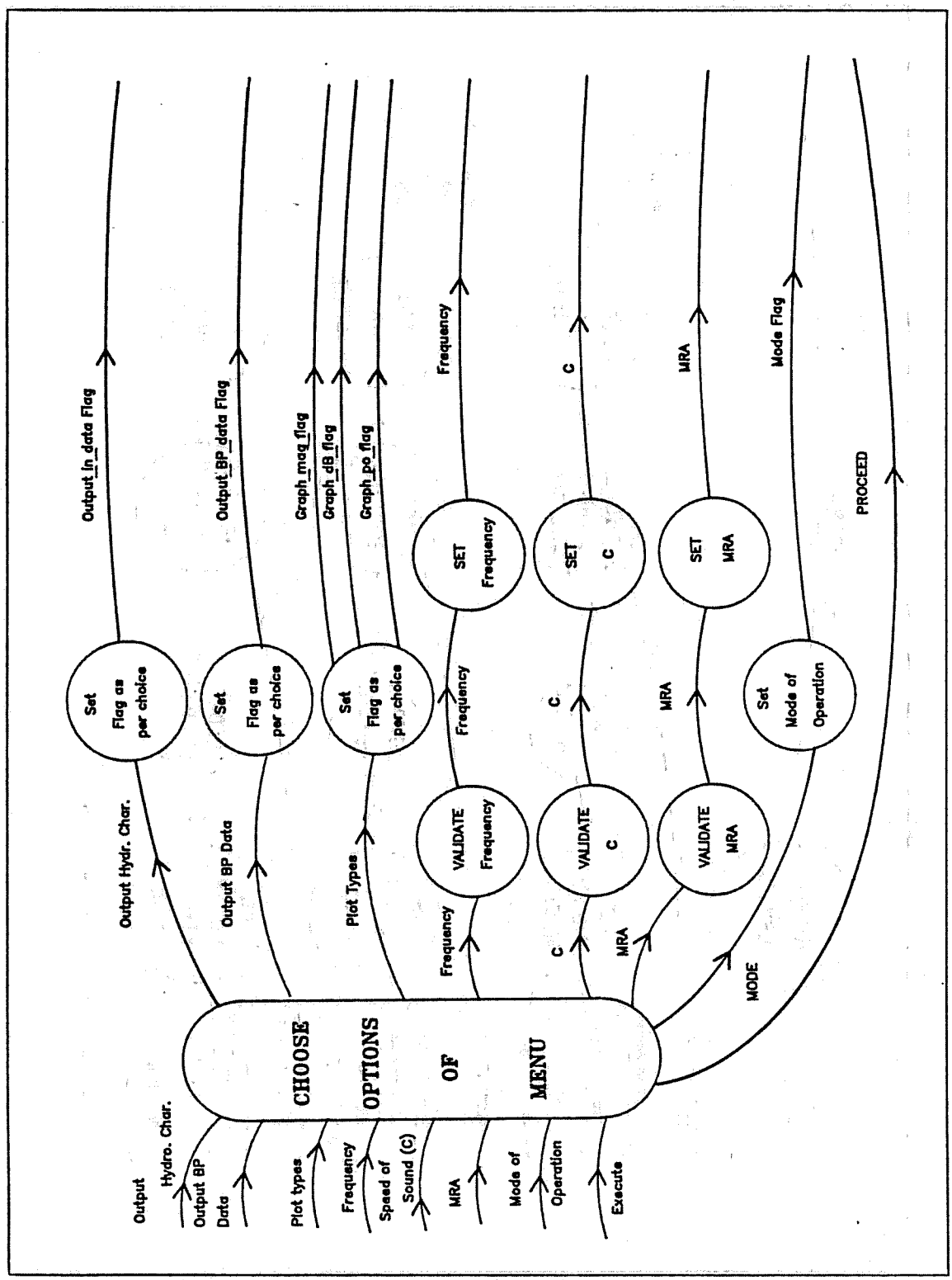

Figure 3.12 DFD of "Choose Beam Pattern Parameters".

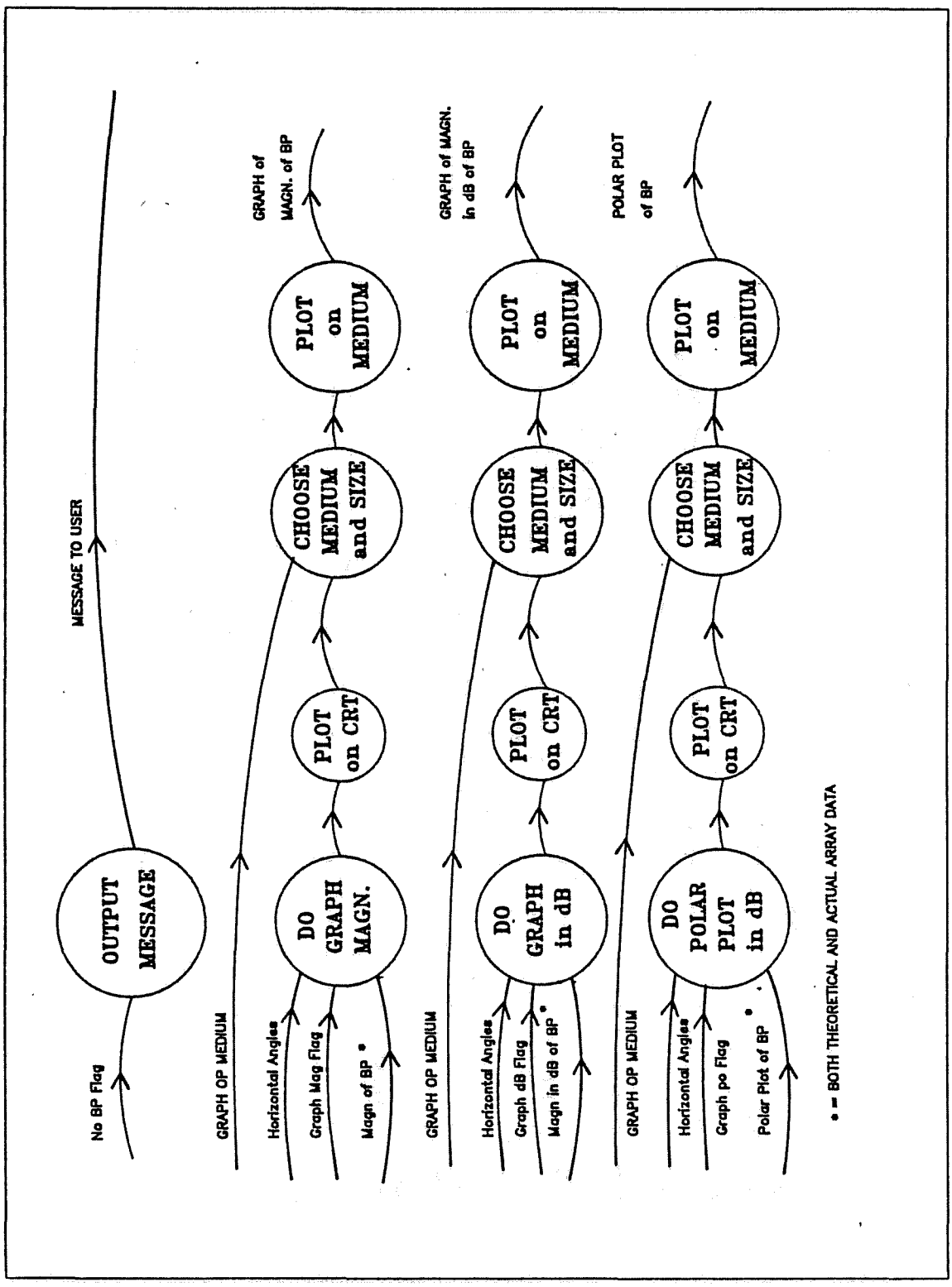

Figure 3.13 DFD of "Choose Graph Output Medium".

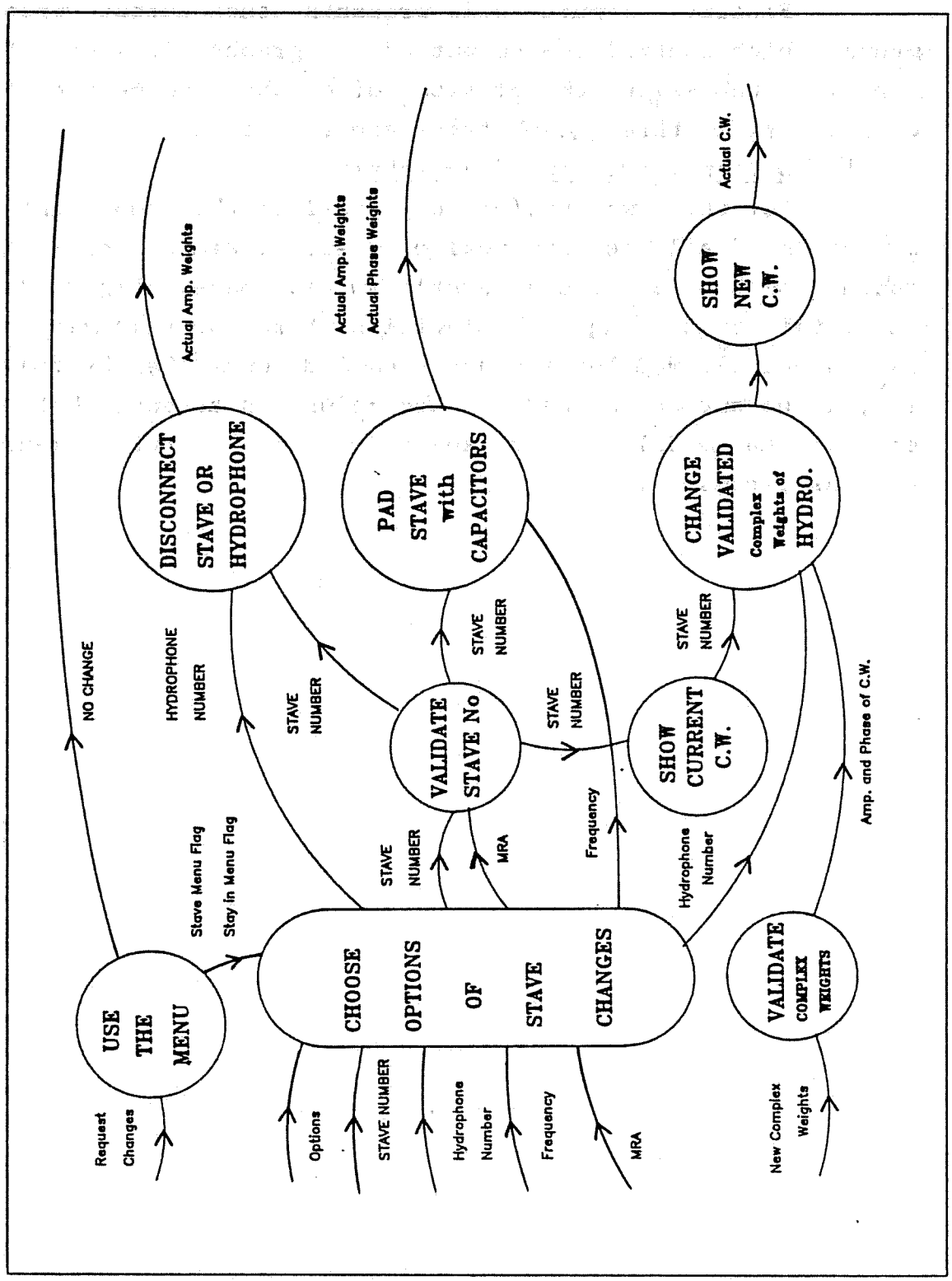

Figure 3.14 DFD of "Modify Individual Stave Status".

Finally, Figure 3.22 presents the "Output graph" module which control the output of all graphs along with the menu for the selection of size of graphs and medium for output. Again three graph types are available.

5. Design : Structured Flowchart

For those who prefer to visualize the sequencing of execution of all the principal modules, a simplified structured flowchart of the main program was made. Figure 3.23 shows this sequencing. No decision boxes or triangles are used since all modules are accessed and controlled by flags, eg., if no output is needed, the module is accessed but the core of the module is not executed and no output is generated as requested.

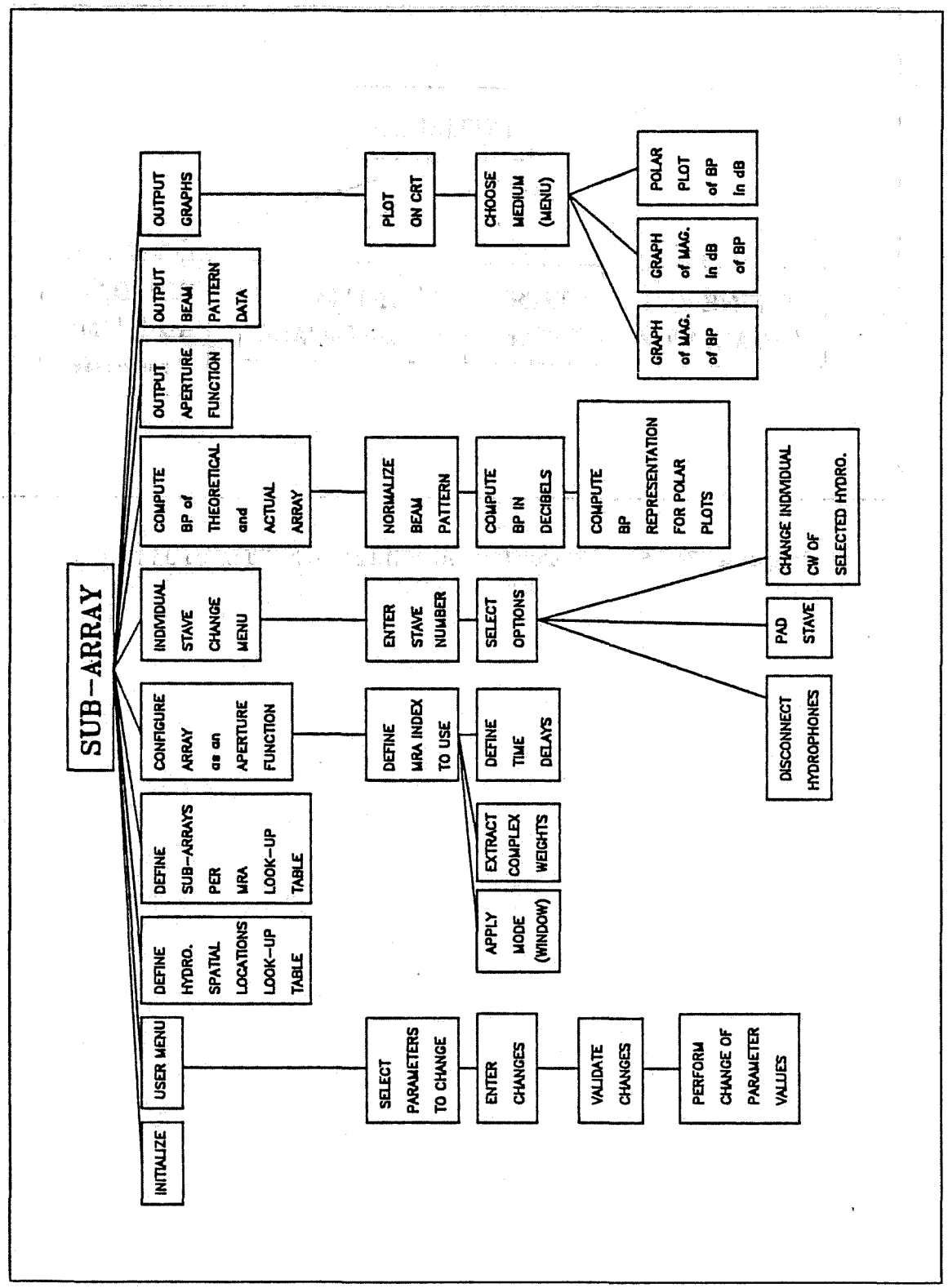

Figure 3.15 Main Structured Chart of SUB-ARRAY.

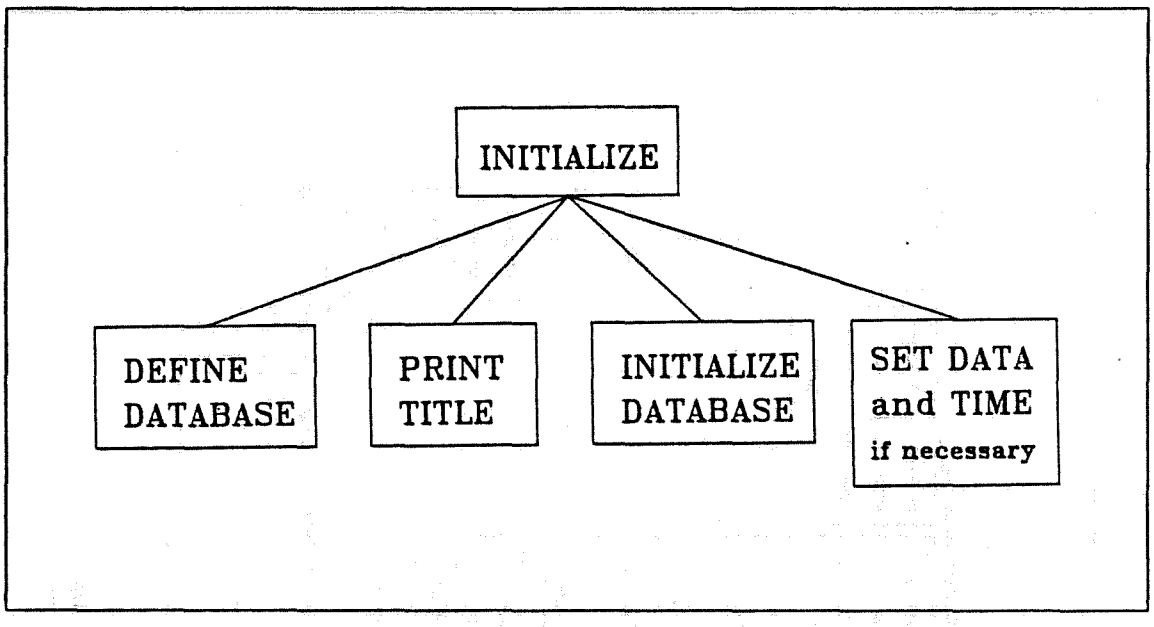

Figure 3.16 Structured Chart of "Initialize".

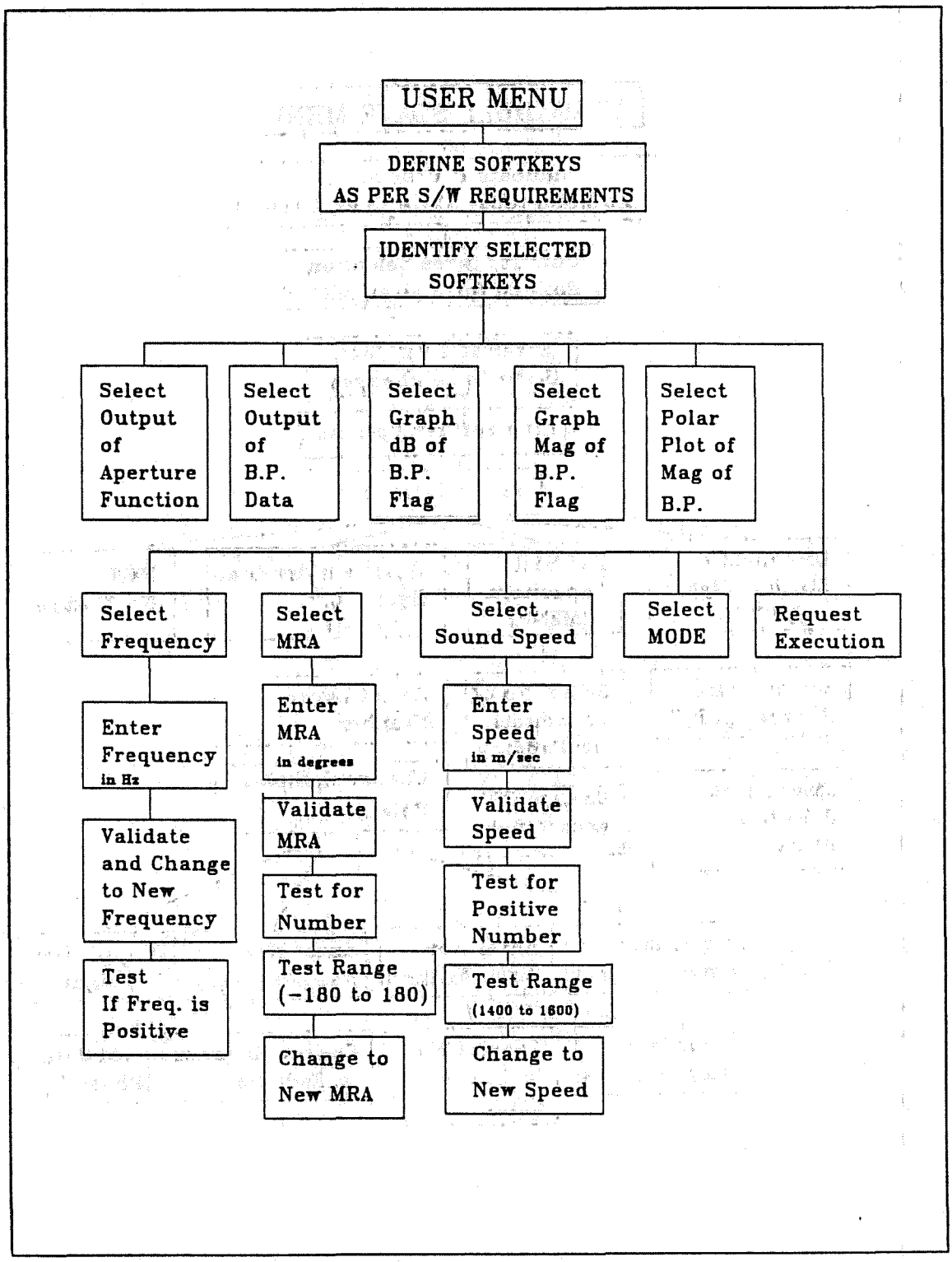

Figure 3.17 Structured Chart of "User Menu".

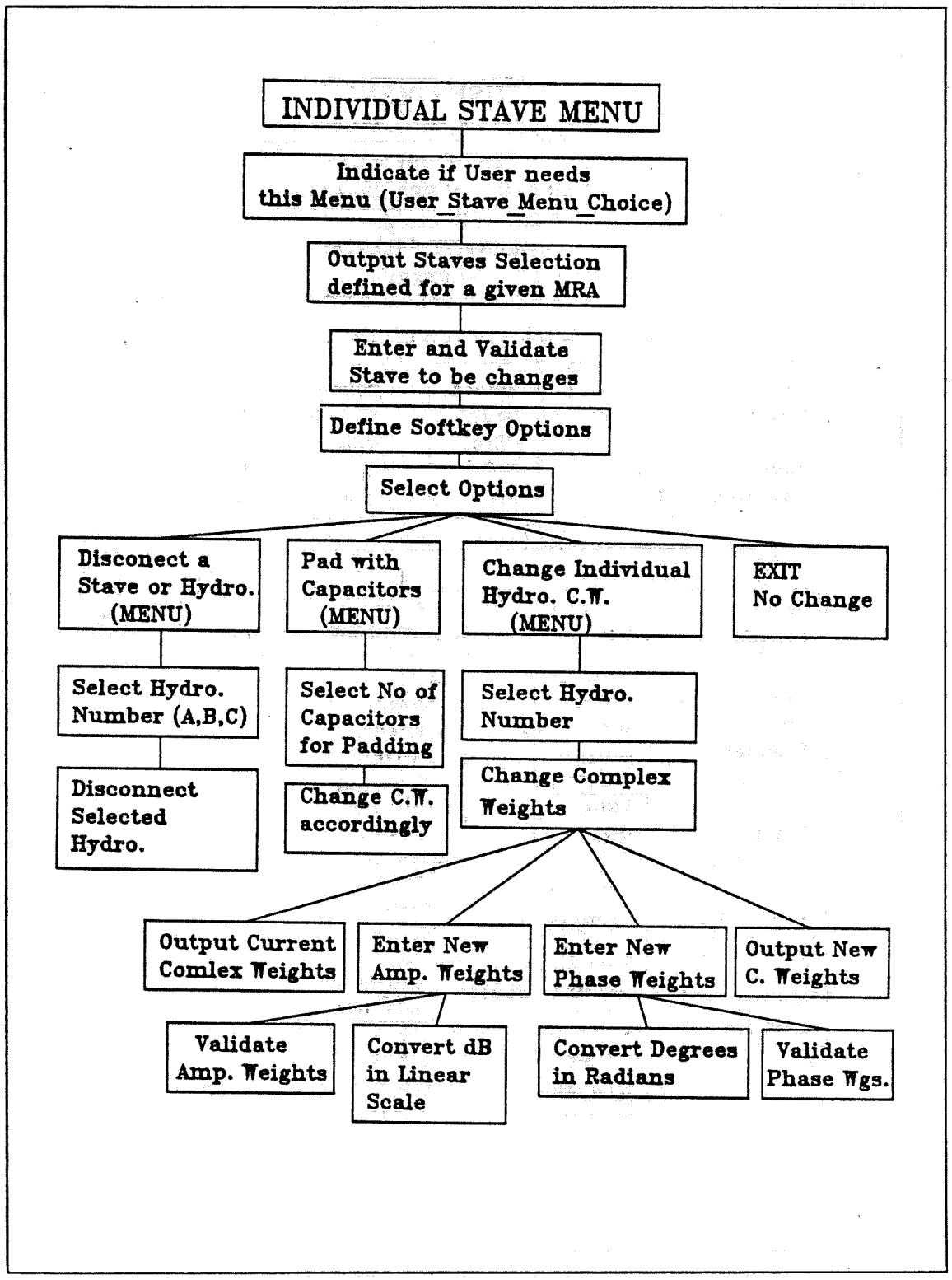

Figure 3.18 Stuctured Chart of "Individual Stave Menu".

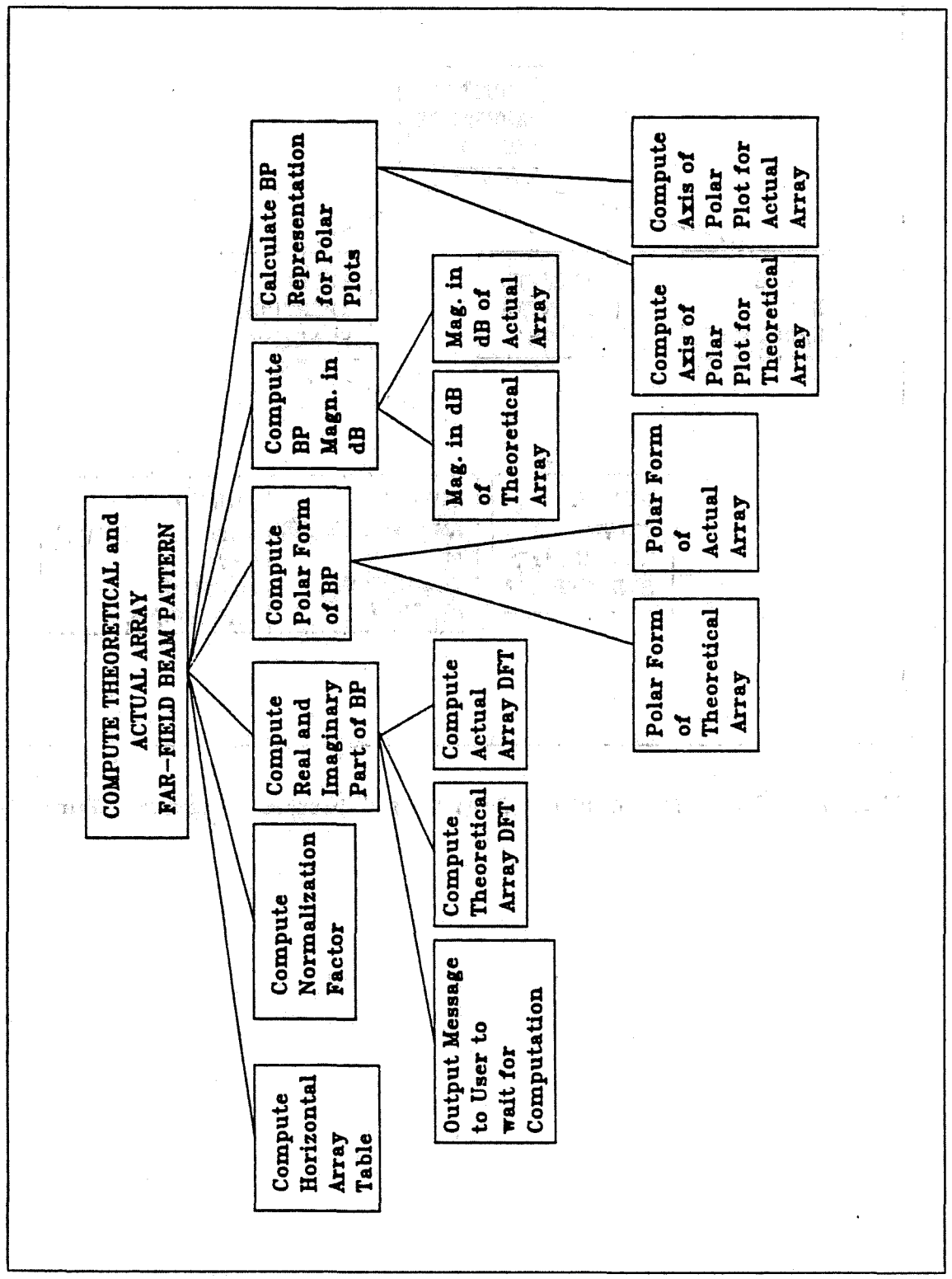

Figure 3.19 Structured Chart of "Compute Beam Pattern".

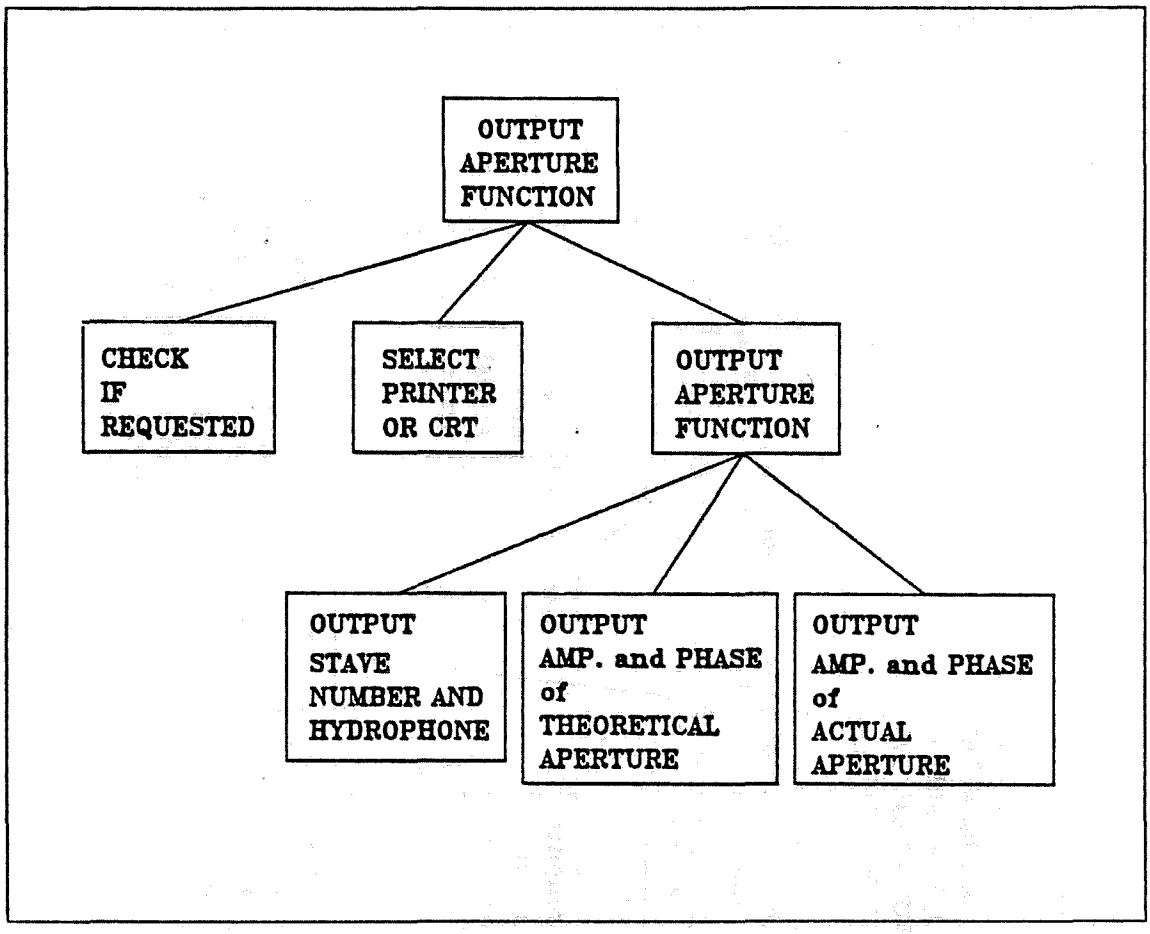

Figure 3.20 Structured Chart of "Output Aperture Function".

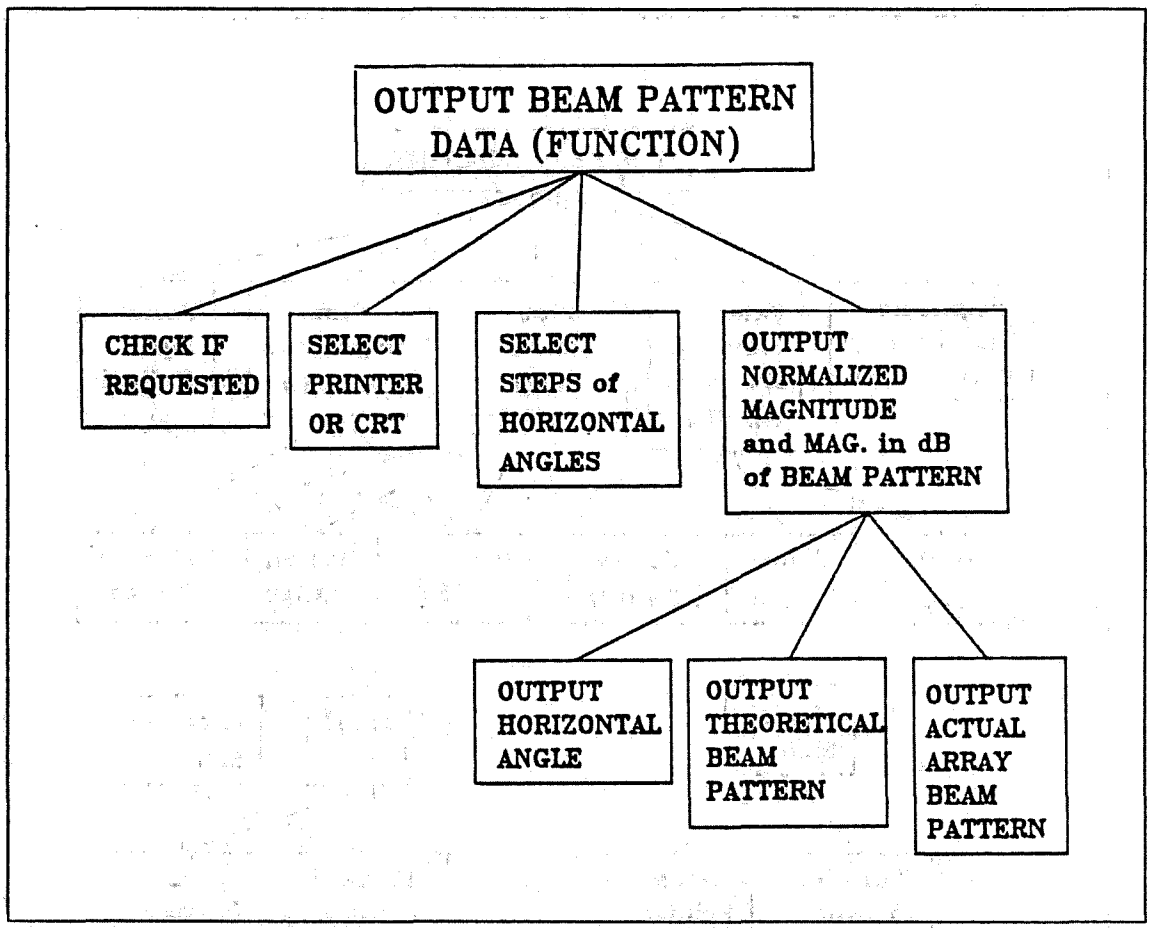

Figure 3.21 Structured Chart of "Output Beam Pattern Data".

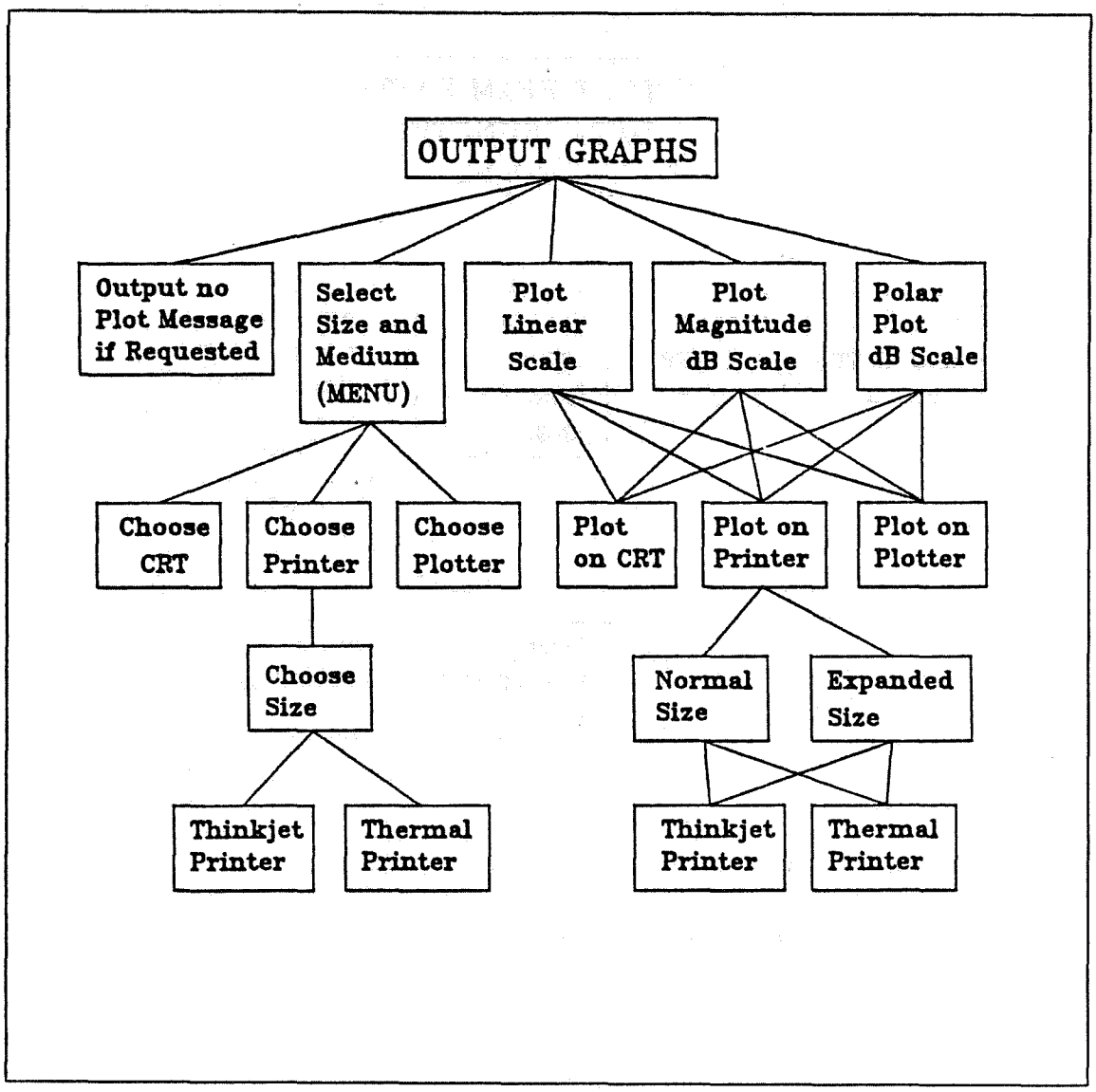

Figure 3.22 Structured Chart of "Output Graphs".

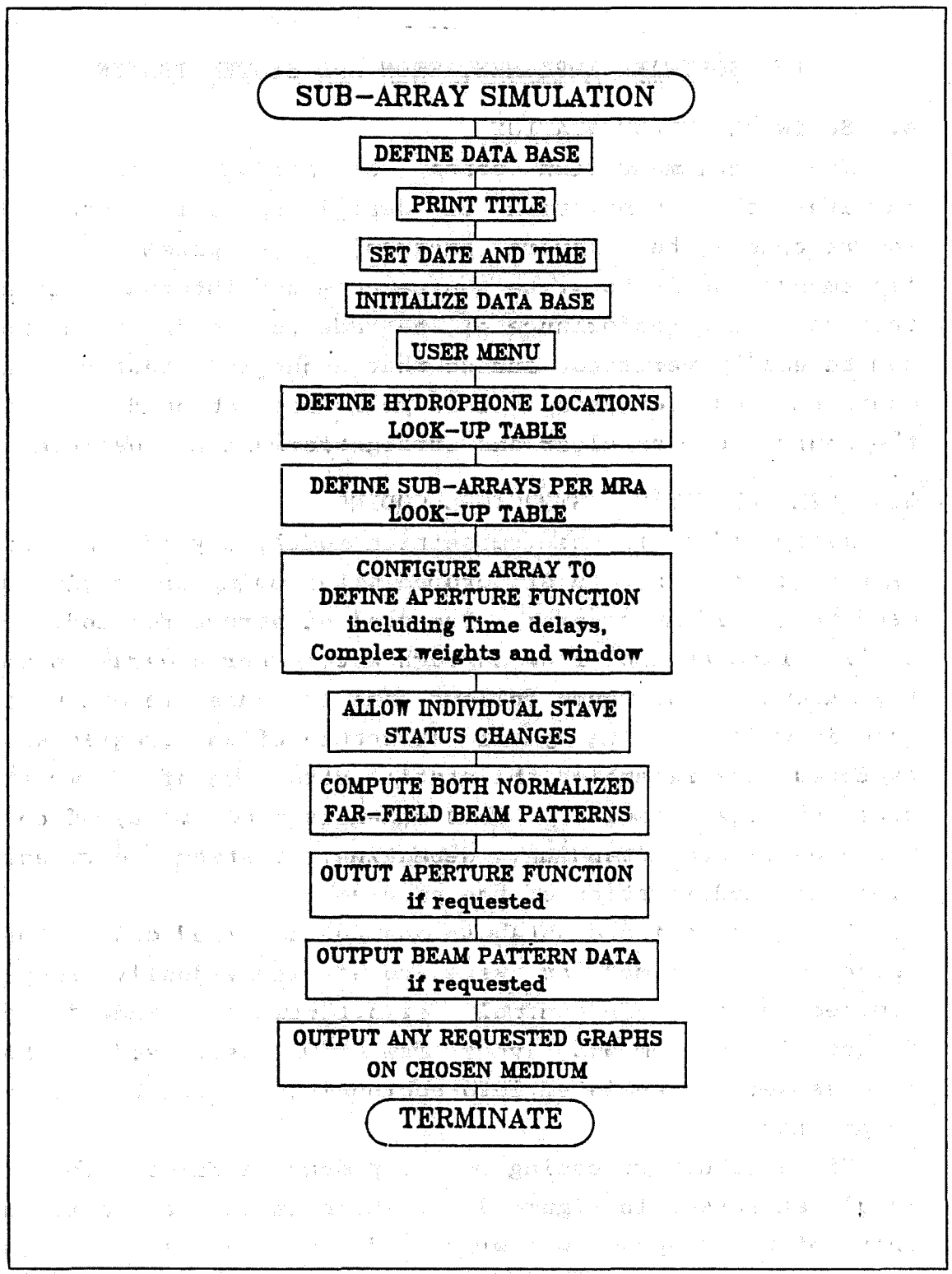

Figure 3.23 Simplified Structured Flowchart of SUB-ARRAY.

IV. SOFTWARE IMPLEMENTATION AND SYSTEM TESTING

A. SOFTWARE IMPLEMENTATION

The implementation phase of software development involves the translation of design specifications into source code (the software program). The primary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing, and modifications are eased. This goal was achieved by making the source code as clear and straightforward as possible.

B. IMPLEMENTATION STRUCTURE CODING

As specified in the subsection modularity of the software requirements, a structured programming or coding was used to write the program. The goal of structured coding is to linearize control flow through a computer program so that the execution sequence follows the sequence in which the code is written. The dynamic structure of the program as it executes then resembles the static structure of the written code (the text itself). This enhances readability of code, which eases understanding, debugging, testing, documentation, and modification of the program.

Using the defined database and the detailed design specification, all modules were coded individually using a limited number of control structures and indentation. Except for the menus, very few GOTOs were used. Most modules were translated into subroutines, functions, and subprograms.

Since structure coding is a top-down technique, the main module expressed in Figure 3.15, which is the main structure chart of the program, was written first. The calls to lower sub-modules were temporarily implemented as "stubs", which

are dummy routines. Later on, these stubs were replaced by the actual sub-modules. The stubs allow the work on top modules then later on lower ones. Therefore, the "calling" modules are coded before their corresponding "called" modules (subordinates). This allows integration of the program to begin early in the cycle of implementation. After individual coding, each module was then unit tested and debugged. Finally, it was incorporated into the program, which was then tested again.

During implementation, redundant lowel level modules, such as testing an input line for a positive number, were eliminated after the calling modules of these lower modules were implemented and tested.

The majority of the routines are specialized and are called by only one module or routine. Additionally, their size was kept to a reasonable value as specified in the design requirements. However, some exceptions were made when clarity was necessary or when the code was simple enough to follow, e.g., the graphs.

Additional code was implemented in order to facilitate debugging. This code generally consists of printing statements of important parameters and control variables. They are used for tracing the execution sequence of a routine and for validation. This code was then commented out, so that the user is not confused by its presence. Schneider and Bruell [Ref. 17: Chap. 9] present a variety of built-in debugging aids (also called debugging instruments).

Finally, looking at the listing, some specific coding standards were used:

1. Use of control structures;
2. All array definitions show the range of their index, e.g., Alpha{1:100};
3. Each loop has its step of counting specified for every case;

4. All end-if statement are commented by the condition of the if;
5. Meaningful variable names are used; and
6. The nesting depths of the program constructs is five or less spaces in normal circumstances.

C. IMPLEMENTATION - UNIT TESTING

After a module (or a unit) is coded, it must be tested separately. This approach is because the proof that a program meets its specification is by partitioning it into smaller modules and proving that each of them meets its intended purpose. The proof is then completed by the unit testing.

Extensive testing of a module is another means of showing that it is correct. Unfortunaly, testing can only reveal the presence of errors, never their absence. To be effective, testing must be done systematically. One approach is to devise sets of test data, that exercise every control path in the module. If it is not practical to do them all, at least the principal ones should be done.

The following cases were used for testing:

1. The Boundary cases: These are the data sets that fall at the very extremes of the legal range of data or fall exactly at the crossover point from one case to another, e.g., choosing stave 1, 26, and 52 for changes;
2. The Null case: the module should work correctly for the degenerate case of "no entry" ,i.e., a null, and a value of zero; and
3. The Illegal case: this is done to test the robustness of the module. This is particularly important for input modules such as the menus. For all illegal cases, no matter how pathological, the module should do something meaningful instead of terminating abnormally. As one final test, the following points were tested:

1. A text of illegal size;
2. An alphanumeric string instread of numeric, e.g., ABC800 or 800 Hz instead of 800;
3. Negative number for positive required data, e.g. -500 Hz; and
4. Real value data instead of integer, e.g., stave 26.75 instead of 26 or 27.

After debugging, all modules successfully passed the unit testing phase.

For every routine, a structured walkthrough was performed. A walkthrough is an in-depth, technical review of some aspect of the software system. It is helpful to discover logic errors and to study the routine environment. Many errors were found that way and corrected before integration. Finally, in some case, hand-simulation was done to help locating problems by "playing" the role of the computer and executing the routine step by step.

D. IMPLEMENTATION - DOCUMENTATION

Program documentation for a software product can be very extensive and covers the entire life-cycle of the product. It was developed continually during program design and implementation. In the particular case of this thesis, only few forms were prepared.

Two basic divisions are used: internal and external documentation. The internal documentation includes standard proloques in the source code as described in Chapter 3 and in internal comments embedded in the source code. The external documentation consists of a Users Manual (see Appendix C) and this thesis. Additionally, the author kept a program notebook which provides the organization of the work activities, early development and results, early specifications, the history of the program, testing results, personal notes and comments, etc. All pertinent documentation useful during the development phases, such as technical

articles, monthly reports, meeting minutes, etc, were also kept in the program notebook.

Except for the user manual which is a separate entity, the system definition, software analysis, requirements specifications, design specifications, and program testing are parts of this thesis. No individual manuals were made for these phases.

The user manual is provided in Appendix C. It is meant solely for end-users. Since the inner workings of the program are immaterial to their needs, the user manual (or quide) concentrates almost exclusively on the input/output characteristics of the program and presents a general overview of what it does. The following features outline the sections of the user manual:

1. Introduction:
   a) Product overview with a brief non-technical description of what the program does;
   b) Terminology and basic features;
   c) Summary of outputs;
2. Getting Started:
   a) Start up;
   b) Input, menus, and options;
   c) Sample runs;
3. Advanced features:
   a) Menu for individual stave changes;
   b) Linear graph and expanded graph; and
4. Program limitations.

The subsection, modularity, of the section software requirements, descibes the required minimum internal documentation in the program source code. Each subroutine, function, and subprogram contains a proloque. Additionally, the program contains a larger proloque at the beginning which also contains a section "description" providing the details of the program.

The program is fully commented. A special attention was made to explain the database in details. Visual dividers in the form of line of asterix or dashes were used to highlight the subsections of a routine and to enhance clarity. Blank lines, borders, and indentation were also used.

Each use of the terminators "END" and "RETURN" is commented to indicate the other half of the control structure, unless a very few statements separate it from the opening control word.

Furthermore, particular attention is made to be sure that comments and code agree with one another, and with the requirements and design specifications. Finally, in order to minimize the need for internal comments in the executable portions of a routine, structure programming constructs, good coding style, and meaningful identifier names were used.

E. IMPLEMENTATION - COMPILATION AND LINKING PROCEDURES

HP BASIC 3.0 language system or any higher version can be used to run the main program "SUB_ARRAY". With the HP BASIC language system, the language extensions of Table III of Chapter 3 are required to execute the program. The steps are, first, load the HP BASIC 3.0 language system, then all the extensions, and then the main program. Pressing the "RUN" key cause the program to be executed. No compilation is required since it is interpreted.

If the Inftotek FP210 Floating Point Coprocessor is used to accelerate the speed of computation, its corresponding routine must first be loaded, like any ordinary extension, before loading and running the main program. Its execution is transparent to the user and requires no change to the software.

Without the coprocessor, the portion computing both theoretical and actual array beam patterns required approximaly forty minutes of CPU time. However, with the coprocessor, this time is reduced by a factor of ten.

Furthermore, the Infotek's Basic Compiler can be used to further reduce processing time by a factor two, i.e., both beam patterns are computed in less than two minutes. The compiler is loaded like an extension. The compiler must match the identification number of the electronic board of the coprocessor. When the compiler is used, the floating point processor routine is not required since it is already part of the compiler. Before execution, after the main program is loaded, simply type "COMPILE" to compile the program. This operation requires about one minute, but needs to be done only once, i.e., as long as the compiled program is in memory. The program can then be executed by simply pressing the key "RUN". The use of the compiler requires much more memory than the standard size, since both source (uncompiled) and object (compiled) codes are kept in memory, along with the compiler itself.

No linking to any other programs is required for the program "SUB_ARRAY".

F. IMPLEMENTATION - DATABASE DEFINITION AND INITIALIZATION

Appendix B provides the complete details of the database and the data stores, along with a listing of the variables. Two subroutines were created for the database, one for its definition, and one for its initialization. HP BASIC initializes all arrays, real, and integer variables to zero, and all string to null when they are defined. This last feature greatly shortens the initialization routine.

G. IMPLEMENTATION - MENU

As described in the requirements and the design sections, three main menus were implemented. They all use softkeys, the ten special keys on top of the keyboard, which allow quick entry into the program. They are interrupt driven and easy to program.

All menus, including the three sub-menus of the menu "Individual stave changes", are constructed the same way. The following components are usually parts of a menu subroutine:

1. Initialization portion;
2. Menu choices with the appropriate comments displayed on the CRT;
3. Softkey definitions;
4. Softkey activation;
5. Wait for a key loop;
6. List of functions for every defined keys;
7. A routine for unused keys which usually consists of a short tone output; and
8. A close-up section that clears the screen and deactivates the softkeys, before exiting the menu.

The function to be performed by each softkey was implemented inside the subroutine, even if it may simply consist of calling another subroutine. The key functions are all accessed by a "GOTO" statement inside the module (a subroutine here), since this is the only possible way to program the softkeys. All called labels of the GOTOs are inside the module defining the menu. Therefore, a menu module or routine is still defined as a single-entry, single-exit module as required by the program specifications.

When the softkeys are activated, a short list of key definitions appears at the bottom of the screen using the alternate screen mode (black on white instead of white on black). The subroutine, softkeys definition, defines these labels. The user can use both the list at the bottom of the screen or the detailed menu on the upper portion of the screen to make a choice. When the softkeys are deactivated, the short list at the bottom of the screen disappears.

The necessity of defining the called labels and functions of all the ten possible GOTOs of the softkeys inside a

menu, make the size of a menu subroutine much larger than the proposed size of a standard subroutine. Each of the functions and labels was clearly identified and packaged like a small routine with only one entry point and one exit point.

The implementation of the menus required the creation of many specialized modules, routines, and functions. Particularly, two functions are used by almost all menus. One test checks if the passed string of characters contains a positive number, the other does the same for any number. They are named:

1. Is_string_pnum; and
2. Is_string_num.

H. IMPLEMENTATION - GRAPHIC PACKAGES

As shown in the design specifications and the requirements, three graph types are generated. The subroutine "Graph_output" controls which graphs are to be generated, uses subroutine "Gph_size_option" to decide what size and on which medium the graphs are to be generated (on an individual basis). The following three subroutines can then be called to generate the required graphs:

1. Plot_magn_bp;
2. Plot_db_bp; and
3. Plot_po_bp.

These subroutines are all constructed the same way. The procedure for constructing a graph is essentially sequential, where every steps add a new component to the graph, i.e., the frame, the title, the legend, the curves, etc.

Even if each subroutine covers about two pages of listing after documentation, it was decided that its size was acceptable and does not need to be separated into smaller segments. Nevertheless, each subroutine uses the following three subroutines:

1. Insert_bot_line: which insert a line of information at the bottom of the graph. It contains the date, the mode of operation, the frequency, and the speed of sound;
2. Insert_legend: which inserts a legend at the right hand side of the graph. It shows both degraded (actual) and theoretical beam pattern line types; and
3. Graph_xxx_label: where xxx is mag, db, or po depending of which graph is done. It is an extension of the database defining the necessary setting variables of a graph. This subroutine is immediately positioned after the graph subroutine.

For this last subroutine, the variables are only initialized, not defined first, since they are standard variables. Every graph subroutine calls one of these "Graph_xxx_label" subroutines as soon as it starts executing.

The basic structure of a graphic subroutine is:

1. Initialize the required database and setting variable;
2. Activate the graphic mode;
3. Scale the graph;
4. Label the graph;
5. Insert line of information at the bottom of the graph;
6. Insert the legend;
7. Label the X and Y axis;
8. Define the viewport and the window;
9. Plot the grid;
10. Indicate the X and Y steps on the graph; and
11. Draw the theoretical and degraded array beam pattern curves.

The exception is the polar plot, where a polar grid is marked with circles and radial lines. No X and Y labels and

steps are shown on the polar plot; however, the dB scale is marked inside the graph.

For the polar plot, only the visible region around the MRA is shown, i.e., MRA - 90 degrees to MRA + 90 degrees is plotted. The rest of the curve is considered irrelevant to the user. Moreover, the deleted portion of the curve does not represent the real curve since diffraction and shadowing by the body of the submarine is not accounted for in the program.

The next chapter provides a large number of Figures showing the final graphical products.

I. IMPLEMENTATION - LISTINGS

The last Appendix of this thesis provide the complete listing of the main program "SUB_ARRAY" which stand for submarine array.

A small subroutine allows the entry of the current date and time, if the present value is not the correct one. It is executed only once when the program is first run.

During the first stage of development, the beam pattern was computed for an odd-numbered line-array of point sources. An FFT algorithm base two was used and implemented as a sub-program. Its proloque is clear enough to allow a simple use. It was left in the code for future use in any modification of the program. The current version does not use this sub-program.

J. SYSTEM TESTING

System testing is the process of exercising or evaluating a system by manual or automated means to verify that it satisfies specific design requirements or to identify differences between expected and actual results.

It is difficult to describe completely the system test phase in a short and concise manner. Therefore, only some of the highlights will be reported. Two kinds of activities were involved: integration testing and acceptance testing.

In integration testing, a strategy for integrating the components of the software system into a functioning whole was used in the form of top down strategy. Integration testing, testing is done during the integration phase, i.e., it starts with the main routine and a few of its subroutines. The testing done here is very similar to the unit testing (see previous sections); however, more emphasis is put on the interface between the routines. After this top level "skeleton" was thoroughly tested, it became the test harness for its immediately subordinate routines. As mentioned before, top-down design requires the use of program stubs to simulate the effects of lower-level routines that are called by those being tested. When these stubs were replaced by the already tested routines, it was tested in its location. To save on re-execution time, usually many non critical routines were combined before being integration tested. However, all critical routines were tested in the program after integration.

Other aspects of the testing include the decision logic, control flow, recovery procedures, throughput, capacity, timing charateristics and, of course, interfaces of the entire system.

After all the routines or modules of the entire system where integrated as a whole in the form of one software product, acceptance testing was performed. It included tests to evaluate the functionality, robustness, and performance of the system so that it satisfies both requirement and design specifications. Some of the test cases used during unit testing and integration testing were performed. Additional test cases were added to achieve the desired level of functionality and performance.

Some test revealed some errors in the time delays implementation, showing that incorrect time delays produced beam pattern curves that were not pointing in the direction

defined by the MRA. Differences of up to 90 degrees were noticed. This implementation problem was corrected.

Some other persons were asked to used the program to evaluate its use and to detect any deficiency in the implementation from the requirement specifications.

Furthermore, a time analysis was done to evaluate the average execution time of the program and especially the beam pattern calculations. The results were within the requirement specifications.

Chapter five provides some of the results of these tests in the form of tables and graphs for different frequency, MRA, speed of sound, and complex relative sensitivities.

K. VERIFICATION & VALIDATION

System testing is part of the verification of the system. Verification establishes the truth of correspondence between a software product and its specifications. It is therefore done throughout the development phases, but its emphasis was stronger during system testing. It was established that the software product satisfies its specification. The reader may refer to Fairley [Ref. 11: Chap. 8] for more details about verification.

After the whole system or product has been integrated, one more stage can be performed: validation. It establishes the fitness or worth of a software product for its operational mission. Since no experimental beam pattern curves were available for comparison, a formal validation could not be made. However, the implemented conformal array was based on a real array; therefore, the theoretical curves should have three important characteristics:

1. Its main lobe must point in the direction defined by the MRA (bearing);
2. Its side lobes should be smaller than the main lobe by about 10 dB; and

3. With the "DIFF: mode a null or definite minimum should occur at the bearing indicated by the MRA.

The next Chapter provides the results of the test cases done during system testing and acceptance. It shows clearly that the program provides the expected beam pattern curves. The program is then considered validated from a theoretical and partially operational point of view.

V. DISCUSSION AND CONCLUSION

A. ANALYSIS AND RESULTS

This section addresses some of the results obtained with the software program "SUB_ARRAY". Many far-field beam pattern curves were generated. The parameters affecting the beam pattern were varied in a systematic manner. The following lists these parameters:

1. Frequency;
2. Sound speed;
3. MRA; and
4. Complex relative sensitivity (magnitude and phase).

1. Summary of Outputs

A complete set of outputs is provided for a given set of parameters. These parameters are:

1. Frequency = 1000 Hz;
2. MRA = 0 degree;
3. Sound speed = 1402.08 m/sec; and
4. Mode of operation = SUM and DIFF.

Table IV shows a typical table of the complex relative sensitivities (magnitude and phase) for both the theoretical and the actual sub-array. Table V shows a table of the theoretical and actual beam pattern data as a function of the horizontal angle (in degrees) for both normalized magnitude (from 0 to 1.0) and decibels scales.

Figure 5.1 shows a polar plot of the beam pattern in dB where only the "visible" region is shown, i.e., MRA - 90 degrees to MRA + 90 degrees. The polar plot is the suggested graph type for operational use. Figure 5.2 shows a rectangular graph of the beam pattern in decibels as a function of the horizontal angle in degrees. Here, the curves cover the entire graph, i.e., from - 180 to 180 degrees. Figure 5.3 shows a figure similar to 5.2 except

that the beam pattern is on a linear scale from 0 to 1.0. The main lobe is much more significant with a linear scale. Figure 5.4 shows a polar plot of the beam pattern in dB for the same case as in Figure 5.1 except the mode of operation is "DIFF" instead of "SUM".

2. Frequency Dependence

Aperture Theory predicts that at a given speed of sound and bearing (MRA), the main lobe beamwidth increases as the frequency decreases. This was confirmed by the program. Additionally, the number of side lobes increases with frequency. Moreover, as the frequency increases, the side lobes levels increase, i.e., there is less and less difference between the main and side lobes. The shape of the beam pattern is much smoother at low frequency than at high frequency.

Figures 5.5 to 5.9 show the curves obtained for a MRA of 0 degrees with frequencies of 100, 200, 400, 800, and 1600 Hz.

3. Sound Speed Dependence

The implemented time delays are theoretically valid at a speed of 1402 m/sec (4600 ft/sec). Therefore, for a fixed electromechanical implementation (see Chapter 3), the beam pattern should vary for different sound speeds. The reason for this variation is explained in Chapter 2 and is based on the fact that the wavelength varies as the sound speed changes. Indeed Figures 5.10, 5.11, and 5.12 show this variation for three different speed of sound, namely 1400, 1500, and 1600 m/sec using time delays designed for a sound speed of 1402 m/sec

The three curves show that the beam pattern, while pointing at the desired bearing, worsens as the speed of sound differs more and more from the design value. Both the shapes and the side lobe levels change.

TABLE IV

COMPLEX RELATIVE SENSITIVITIES OF THE HYDROPHONES

THE REQUESTED MRA = 0 degrees
THE USED MRA = 0 degrees

THE MODE OF OPERATION = SUM

THE COMPLEX RELATIVE SENSITIVITIES USED ARE

HYDRO. NUMBER M , P	DESIGN AMPL X	DESIGN PHASE X	DEGRADED AMPL X	DEGRADED PHASE X
13 -1	1.00000	0.00000	1.00000	0.00000
13 0	1.00000	0.00000	1.00000	0.00000
13 1	1.00000	0.00000	1.00000	0.00000
14 -1	1.00000	0.00000	1.00000	0.00000
14 0	1.00000	0.00000	1.00000	0.00000
14 1	1.00000	0.00000	1.00000	0.00000
15 -1	1.00000	0.00000	1.00000	0.00000
15 0	1.00000	0.00000	1.00000	0.00000
15 1	1.00000	0.00000	1.00000	0.00000
16 -1	1.00000	0.00000	1.00000	0.00000
16 0	1.00000	0.00000	1.00000	0.00000
16 1	1.00000	0.00000	1.00000	0.00000
17 -1	1.00000	0.00000	1.00000	0.00000
17 0	1.00000	0.00000	1.00000	0.00000
17 1	1.00000	0.00000	1.00000	0.00000
18 -1	1.00000	0.00000	1.00000	0.00000
18 0	1.00000	0.00000	1.00000	0.00000
18 1	1.00000	0.00000	1.00000	0.00000
19 -1	1.00000	0.00000	1.00000	0.00000
19 0	1.00000	0.00000	1.00000	0.00000
19 1	1.00000	0.00000	1.00000	0.00000
20 -1	1.00000	0.00000	1.00000	0.00000
20 0	1.00000	0.00000	1.00000	0.00000
20 1	1.00000	0.00000	1.00000	0.00000
21 -1	1.00000	0.00000	1.00000	0.00000
21 0	0.00000	0.00000	0.00000	0.00000
21 1	1.00000	0.00000	1.00000	0.00000
22 -1	1.00000	0.00000	1.00000	0.00000
22 0	1.00000	0.00000	1.00000	0.00000
22 1	1.00000	0.00000	1.00000	0.00000
23 -1	1.00000	0.00000	1.00000	0.00000
23 0	1.00000	0.00000	1.00000	0.00000
23 1	1.00000	0.00000	1.00000	0.00000
24 -1	1.00000	0.00000	1.00000	0.00000
24 0	1.00000	0.00000	1.00000	0.00000
24 1	1.00000	0.00000	1.00000	0.00000
25 -1	0.00000	0.00000	0.00000	0.00000
25 0	1.00000	0.00000	1.00000	0.00000
25 1	0.00000	0.00000	0.00000	0.00000
26 -1	0.00000	0.00000	0.00000	0.00000
26 0	1.00000	0.00000	1.00000	0.00000
26 1	0.00000	0.00000	0.00000	0.00000
27 -1	0.00000	0.00000	0.00000	0.00000
27 0	1.00000	0.00000	1.00000	0.00000
27 1	0.00000	0.00000	0.00000	0.00000
28 -1	0.00000	0.00000	0.00000	0.00000
28 0	1.00000	0.00000	1.00000	0.00000
28 1	0.00000	0.00000	0.00000	0.00000
29 -1	1.00000	0.00000	1.00000	0.00000
29 0	1.00000	0.00000	1.00000	0.00000
29 1	1.00000	0.00000	1.00000	0.00000
30 -1	1.00000	0.00000	1.00000	0.00000
30 0	1.00000	0.00000	1.00000	0.00000
30 1	1.00000	0.00000	1.00000	0.00000
31 -1	1.00000	0.00000	1.00000	0.00000
31 0	1.00000	0.00000	1.00000	0.00000
31 1	1.00000	0.00000	1.00000	0.00000
32 -1	1.00000	0.00000	1.00000	0.00000
32 0	1.00000	0.00000	1.00000	0.00000
32 1	1.00000	0.00000	1.00000	0.00000
33 -1	1.00000	0.00000	1.00000	0.00000
33 0	1.00000	0.00000	1.00000	0.00000
33 1	1.00000	0.00000	1.00000	0.00000
34 -1	1.00000	0.00000	1.00000	0.00000
34 0	1.00000	0.00000	1.00000	0.00000
34 1	1.00000	0.00000	1.00000	0.00000
35 -1	1.00000	0.00000	1.00000	0.00000
35 0	1.00000	0.00000	1.00000	0.00000
35 1	1.00000	0.00000	1.00000	0.00000
36 -1	1.00000	0.00000	1.00000	0.00000
36 0	1.00000	0.00000	1.00000	0.00000
36 1	1.00000	0.00000	1.00000	0.00000
37 -1	1.00000	0.00000	1.00000	0.00000
37 0	1.00000	0.00000	1.00000	0.00000
37 1	1.00000	0.00000	1.00000	0.00000
38 -1	1.00000	0.00000	1.00000	0.00000
38 0	1.00000	0.00000	1.00000	0.00000
38 1	1.00000	0.00000	1.00000	0.00000
39 -1	1.00000	0.00000	1.00000	0.00000
39 0	1.00000	0.00000	1.00000	0.00000
39 1	1.00000	0.00000	1.00000	0.00000
40 -1	1.00000	0.00000	1.00000	0.00000
40 0	1.00000	0.00000	1.00000	0.00000
40 1	1.00000	0.00000	1.00000	0.00000

TABLE V

THEORETICAL AND ACTUAL ARRAY BEAM PATTERN DATA

THE REQUESTED MRA = 0 degrees
THE USED MRA = 0 degrees

THE MODE OF OPERATION = SUM

The execution time to compute the beam pattern was 885.490020752 SEC

*** COMPUTED MAGNITUDE BEAM PATTERN DATA ***

ANGLE	THEORETICAL		ACTUAL	
degrees	Normalized	dB	Normalized	dB
-180	.15053	-16.45	.15053	-16.45
-170	.03805	-28.39	.03805	-28.39
-160	.03537	-29.03	.03537	-29.03
-150	.05104	-25.84	.05104	-25.84
-140	.01575	-36.06	.01575	-36.06
-130	.08801	-21.11	.08801	-21.11
-120	.09196	-20.73	.09196	-20.73
-110	.05082	-25.88	.05082	-25.88
-100	.03116	-30.13	.03116	-30.13
-90	.08945	-20.97	.08945	-20.97
-80	.12723	-17.91	.12723	-17.91
-70	.12758	-17.88	.12758	-17.88
-60	.21038	-13.54	.21038	-13.54
-50	.34189	-9.32	.34189	-9.32
-40	.42620	-7.41	.42620	-7.41
-30	.42100	-7.51	.42100	-7.51
-20	.31659	-9.99	.31659	-9.99
-10	.31617	-10.00	.31617	-10.00
0	1.00000	0.00	1.00000	0.00
10	.30825	-10.22	.30825	-10.22
20	.31570	-10.01	.31570	-10.01
30	.39467	-8.08	.39467	-8.08
40	.42581	-7.42	.42581	-7.42
50	.33643	-9.46	.33643	-9.46
60	.18888	-14.48	.18888	-14.48
70	.12479	-18.08	.12479	-18.08
80	.12321	-18.19	.12321	-18.19
90	.08713	-21.20	.08713	-21.20
100	.03028	-30.38	.03028	-30.38
110	.04359	-27.21	.04359	-27.21
120	.08020	-21.92	.08020	-21.92
130	.07871	-22.08	.07871	-22.08
140	.01573	-36.07	.01573	-36.07
150	.06156	-24.21	.06156	-24.21
160	.03773	-28.47	.03773	-28.47
170	.06287	-24.03	.06287	-24.03
180	.15053	-16.45	.15053	-16.45

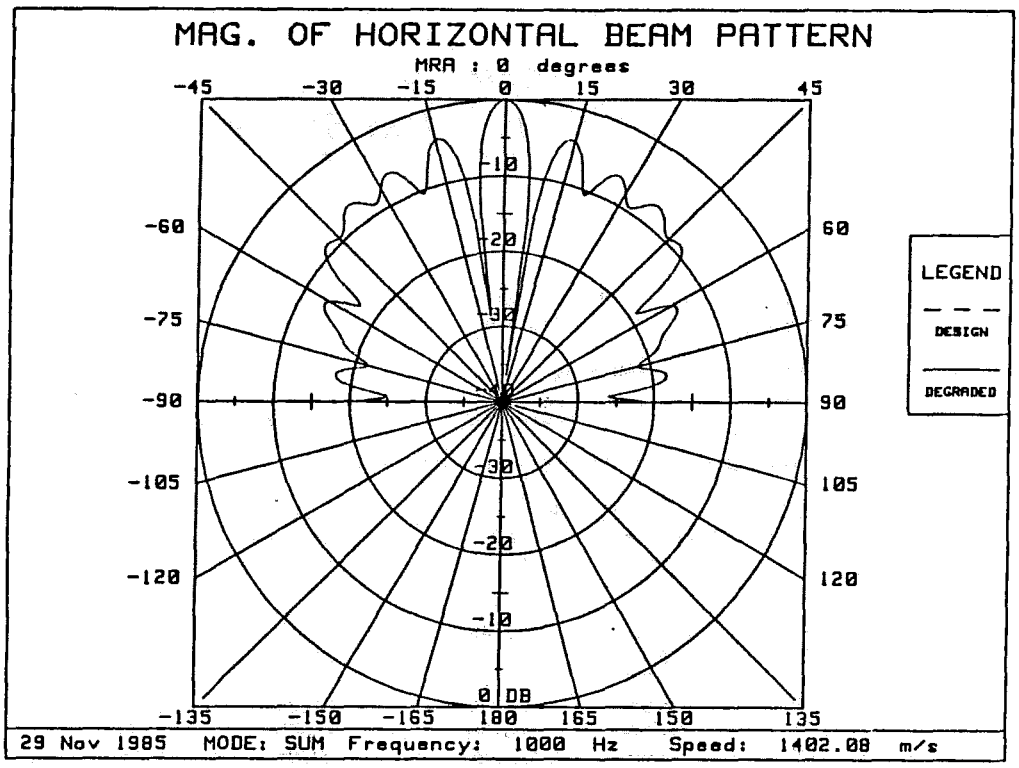

Figure 5.1 Polar Plot of Beam Pattern in dB.

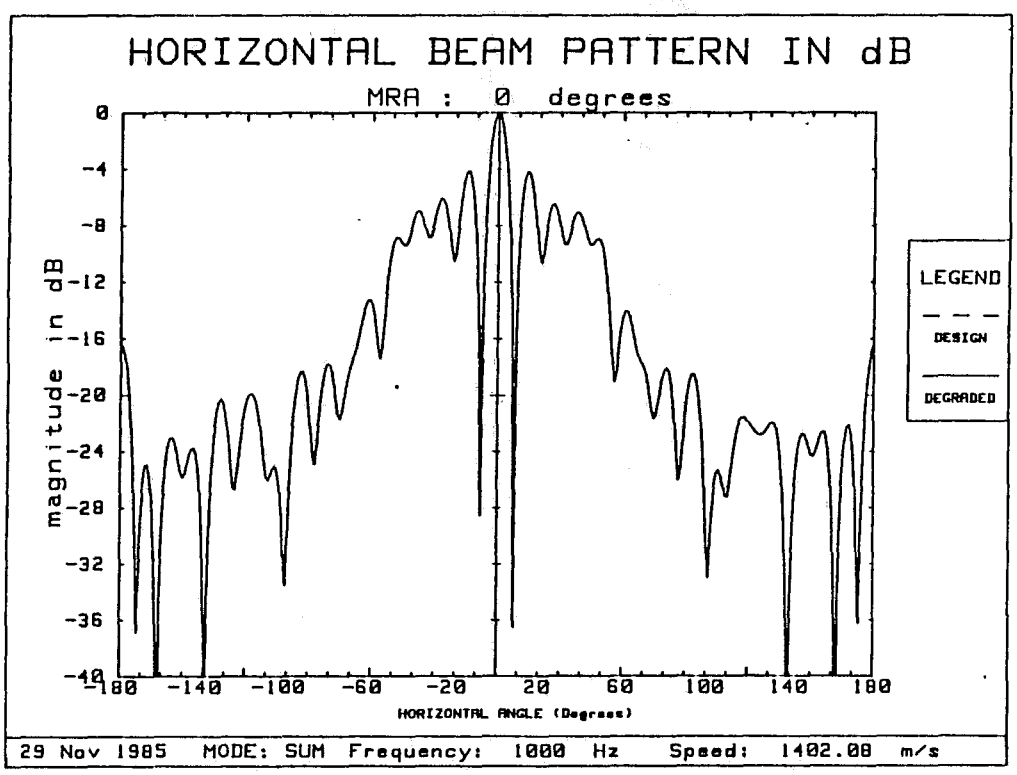

Figure 5.2 Magnitude of Beam Pattern in dB.

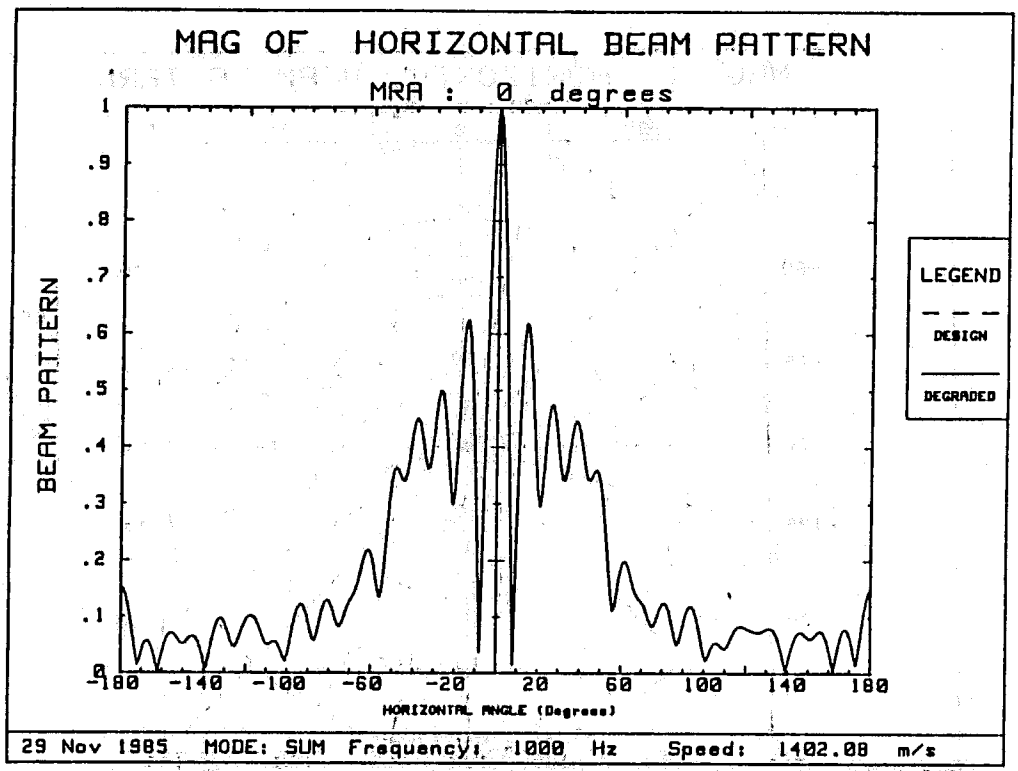

Figure 5.3 Normalized Magnitude of Beam Pattern.

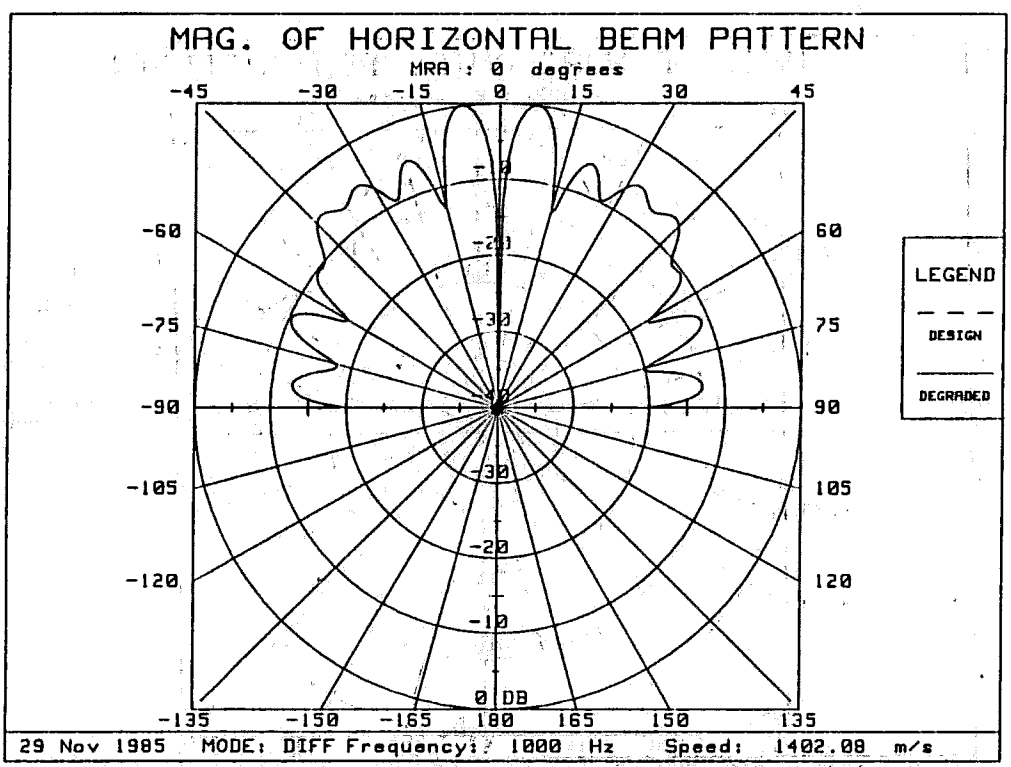

Figure 5.4 Polar Plot of Beam Pattern with "DIFF" Mode.

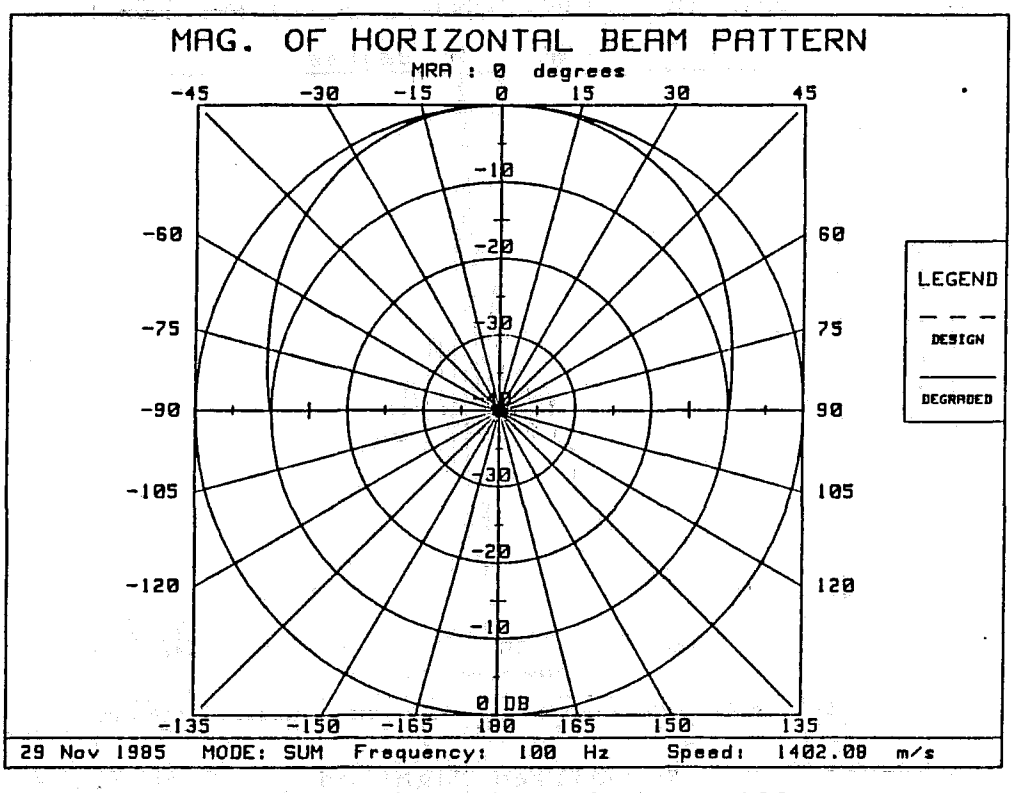

Figure 5.5 Polar Plot of BP at 100 Hz.

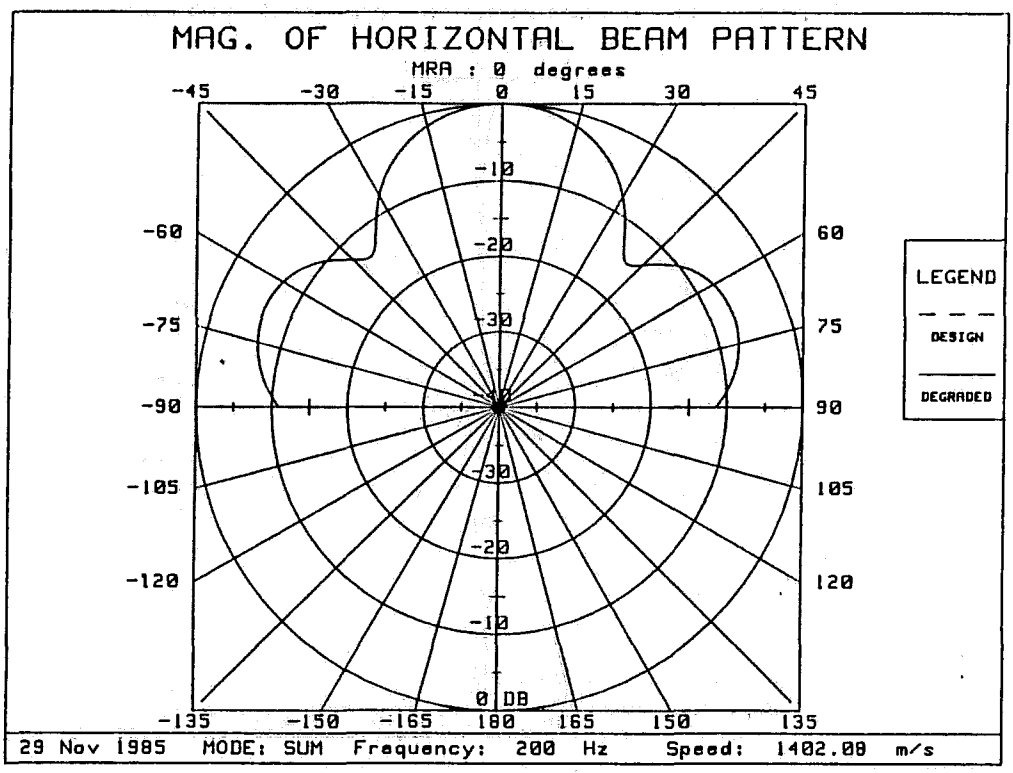

Figure 5.6 Polar Plot of BP at 200 Hz.

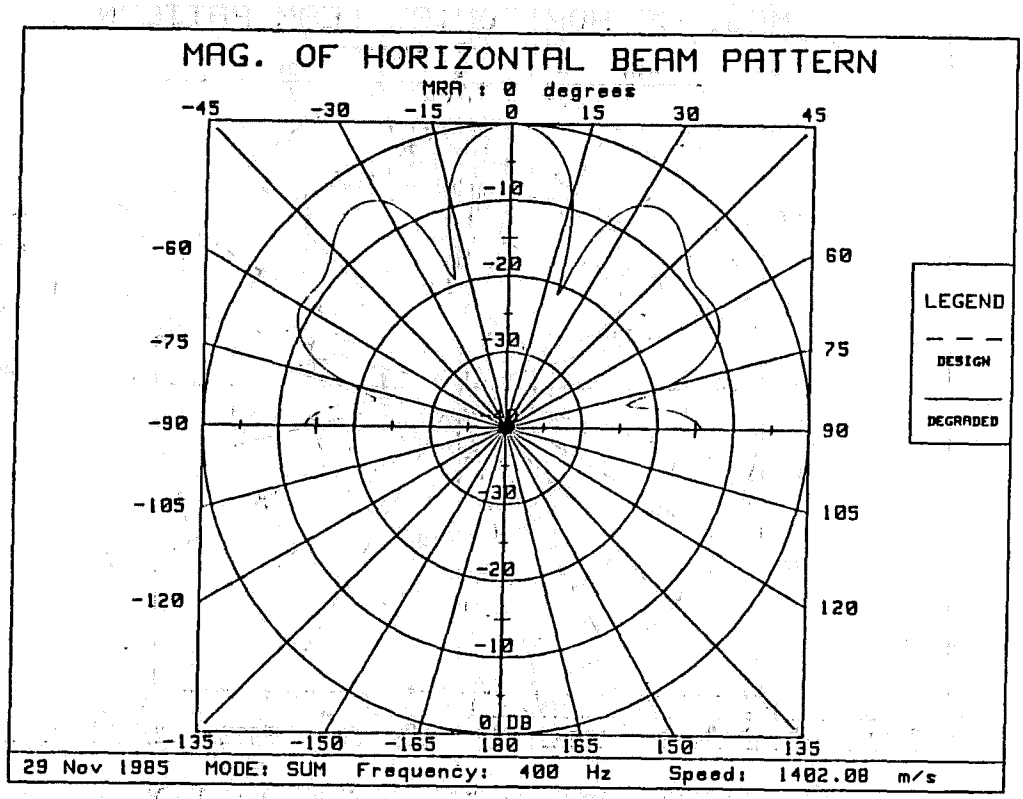

Figure 5.7 Polar Plot of BP at 400 Hz.

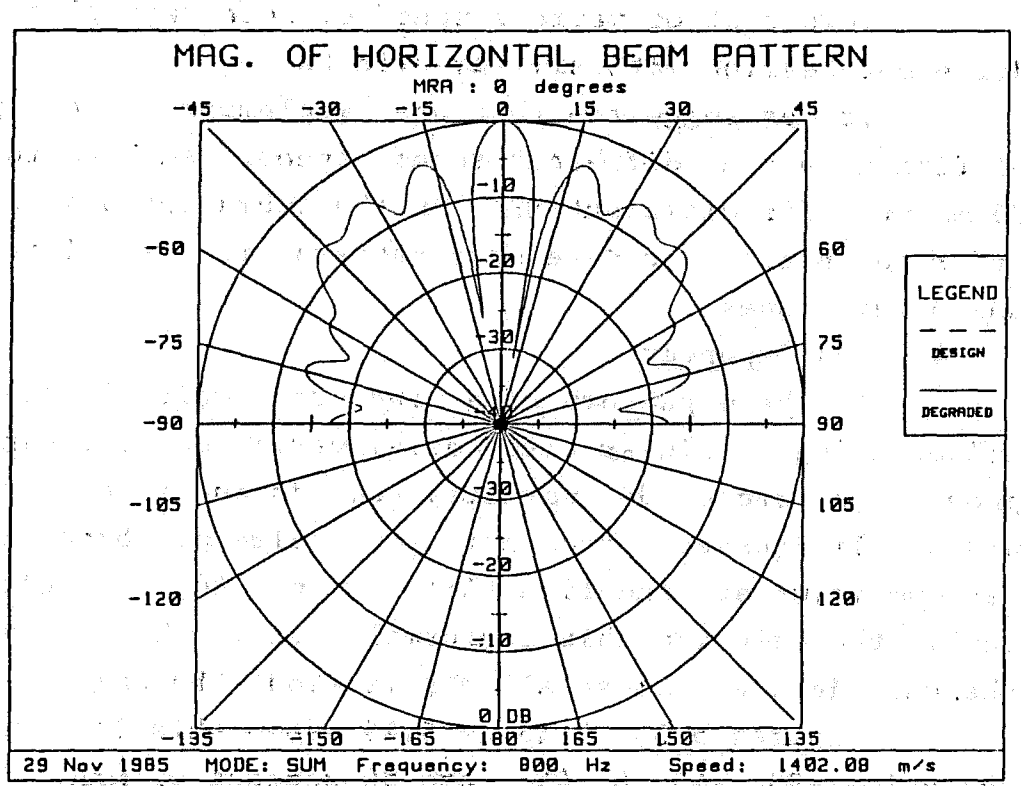

Figure 5.8 Polar Plot of BP at 800 Hz.

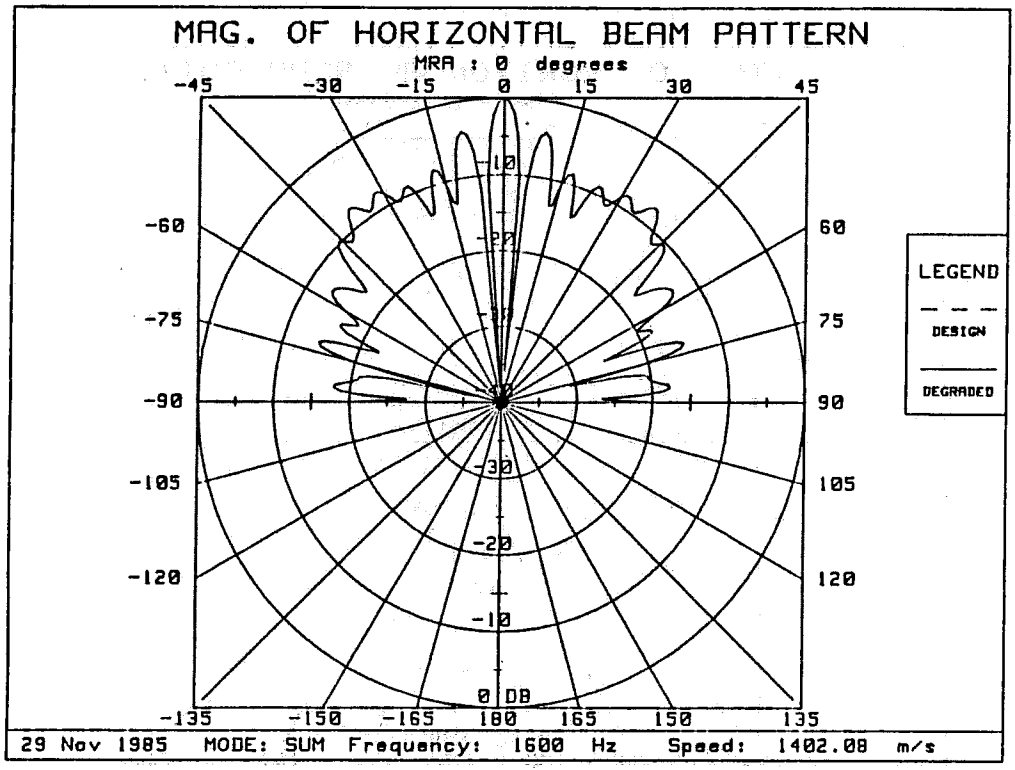

Figure 5.9 Polar Plot of BP at 1600 Hz.

This test of various speed of sound was also done at different bearing with similar effects.

It was suggested that one implement a look-up table of time delays at different sound speed, such as for every 25 m/sec, for different MRA in the submarine beamformer in order to obtain accurate beam pattern with significant and sharp main lobes.

4. MRA Dependence

The beam pattern of a line array which is steered or tilted using a linear time delay variation between hydrophones from broadside (on-axis) towards end-fire, its beamwidth will increase from a minimum value at broadside to a maximum value at end-fire [Ref. 2: p. 64]. This is not really the case for this conformal array. The beamwidth of the main lobe at almost all MRA is about the same.

Figures 5.3, 5.13 to 5.16 show this last point for the value -90, -45, 0, 45, and 90 degrees of MRA. Only at

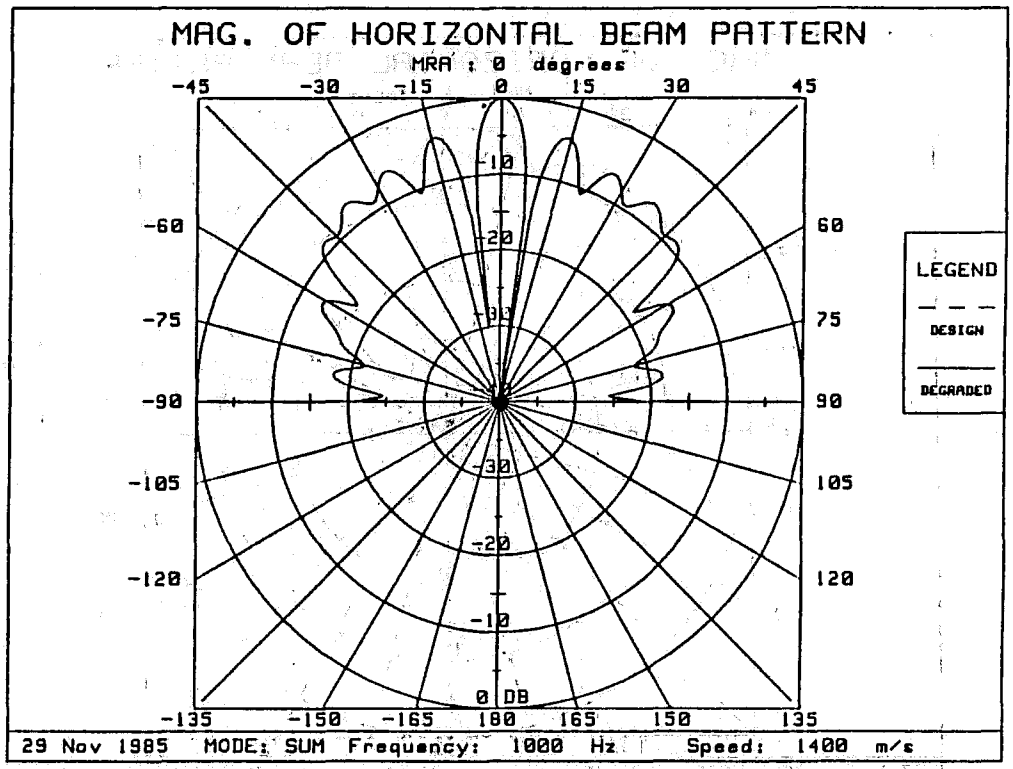

Figure 5.10 Polar Plot of BP at 1400 m/sec.

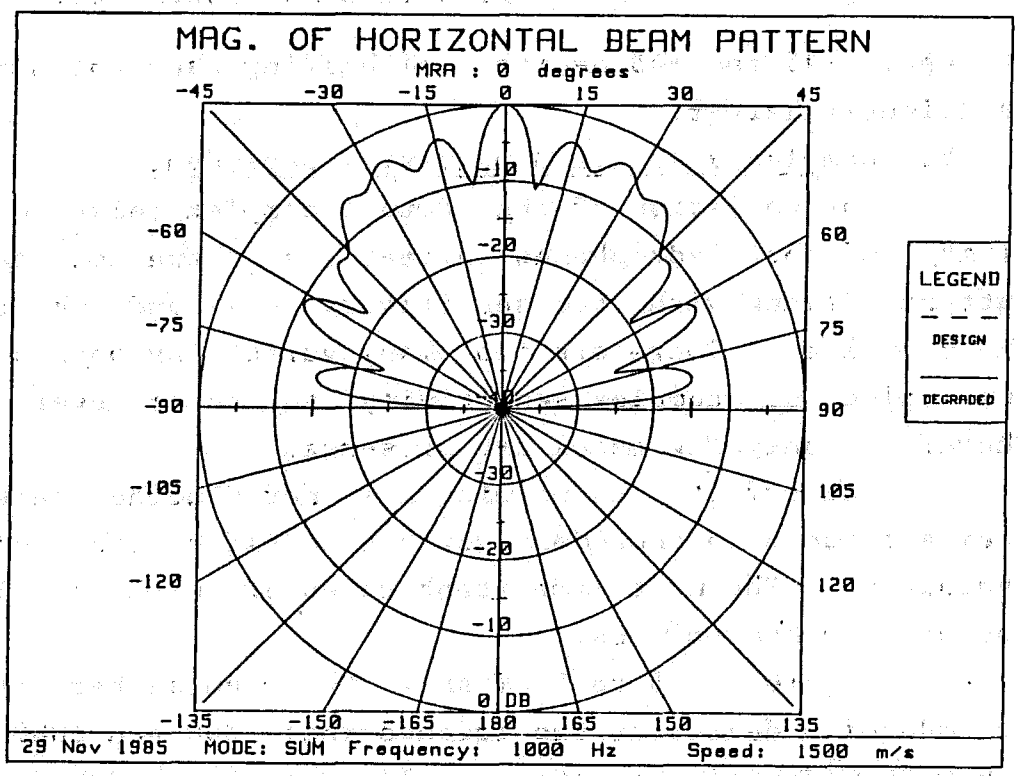

Figure 5.11 Polar Plot of BP at 1500 m/sec.

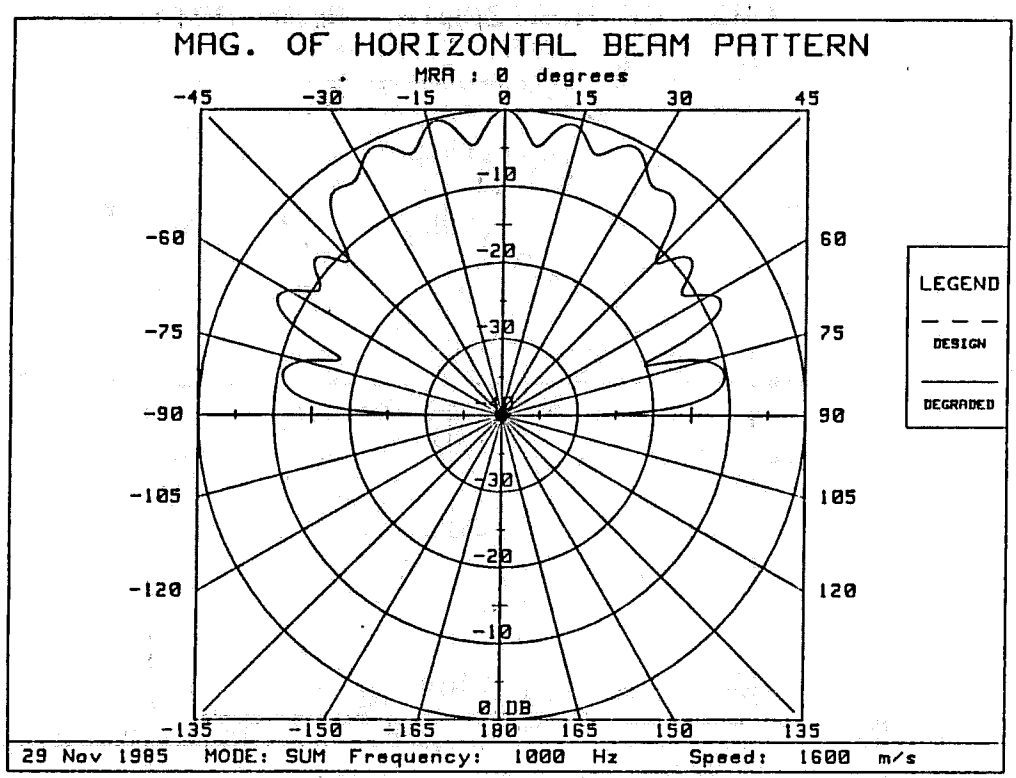

Figure 5.12 Polar Plot of BP at 1600 m/sec.

the angle -45 and +45 degrees of bearing that the main lobe is slightly larger.

5. Complex Relative Sensitivity Dependence

The variation of the actual complex relative sensitivity of the hydrophones affect the shape of the beam pattern, along with the sensitivity level and the bearing (MRA) value. There are too many variations possible with the value of complex sensitivity to cover every case. Therefore, only few cases are presented.

The effects of padding are not studied here since they are too much dependent of the specific situation being evaluated. The most important cases are with the loss of one or more hydrophones.

Figures 5.18 to 5.21 show the beam pattern obtained for MRA of 0 degree at two frequencies: 300 and 1000 Hz and for both modes of operation. The disconnected hydrophones for this case (number 1) are shown in black in Figure 5.17 .

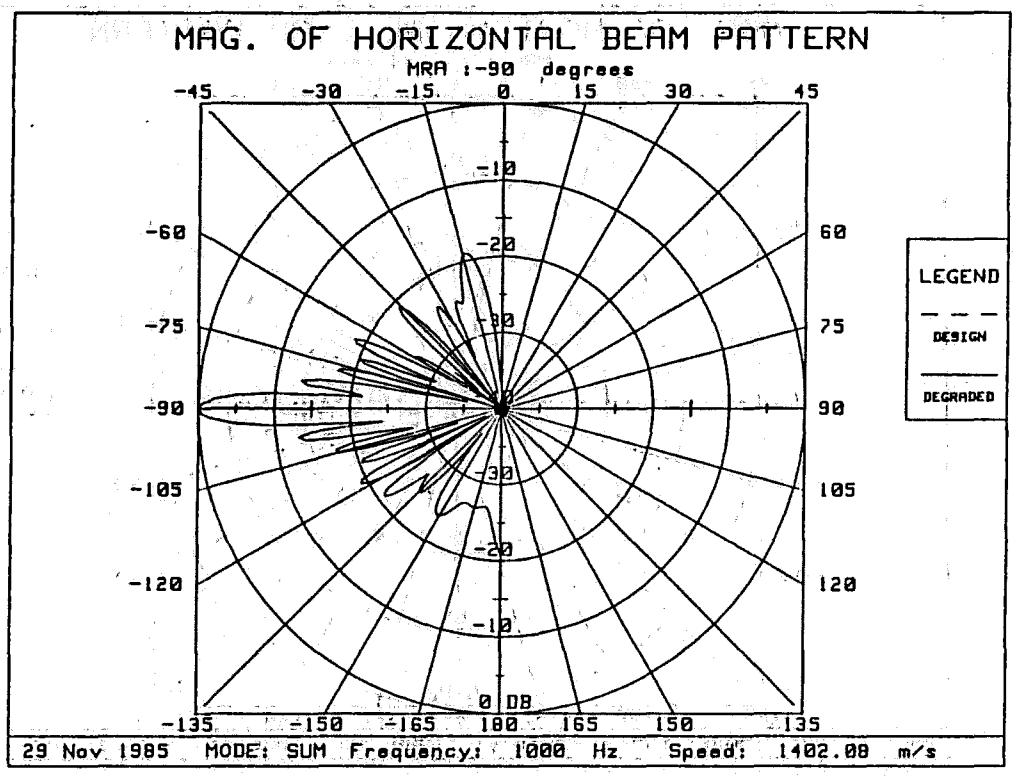

Figure 5.13 Polar Plot of BP at -90 degrees.

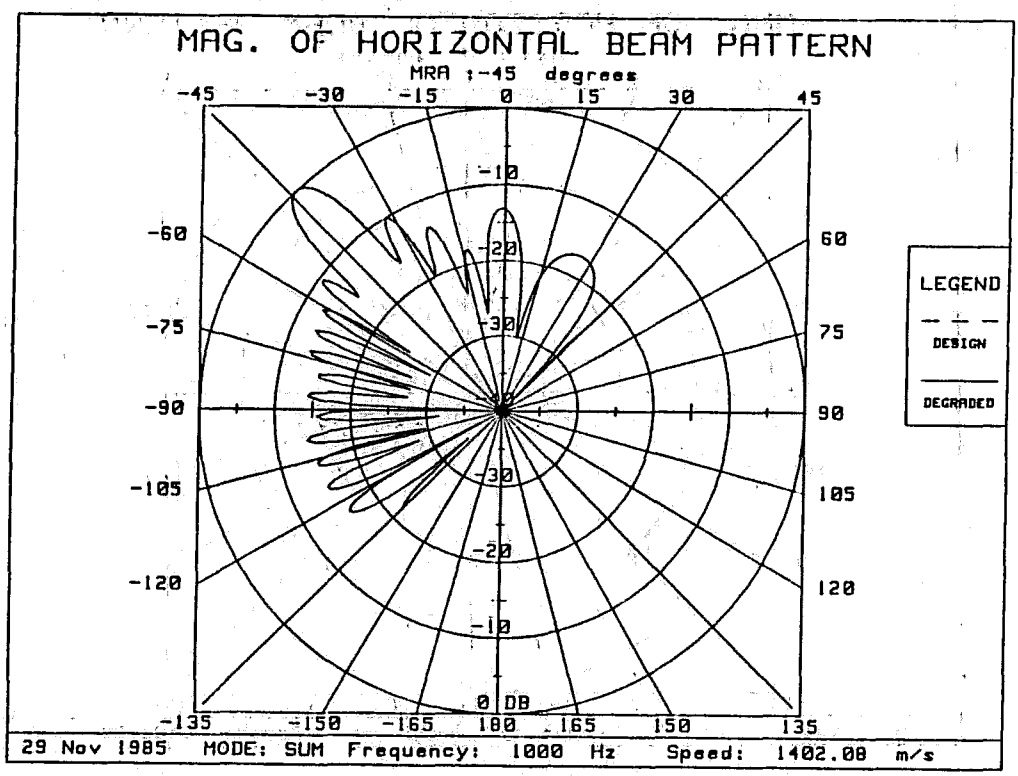

Figure 5.14 Polar Plot of BP at -45 degrees.

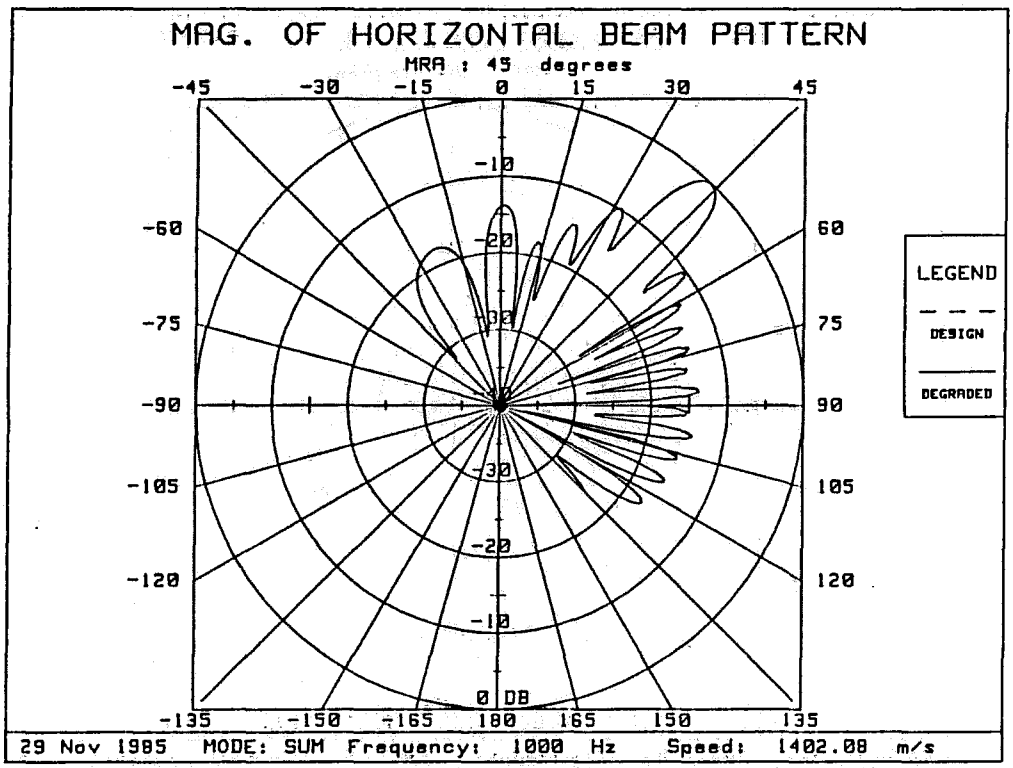

Figure 5.15 Polar Plot of BP at 45 degrees.

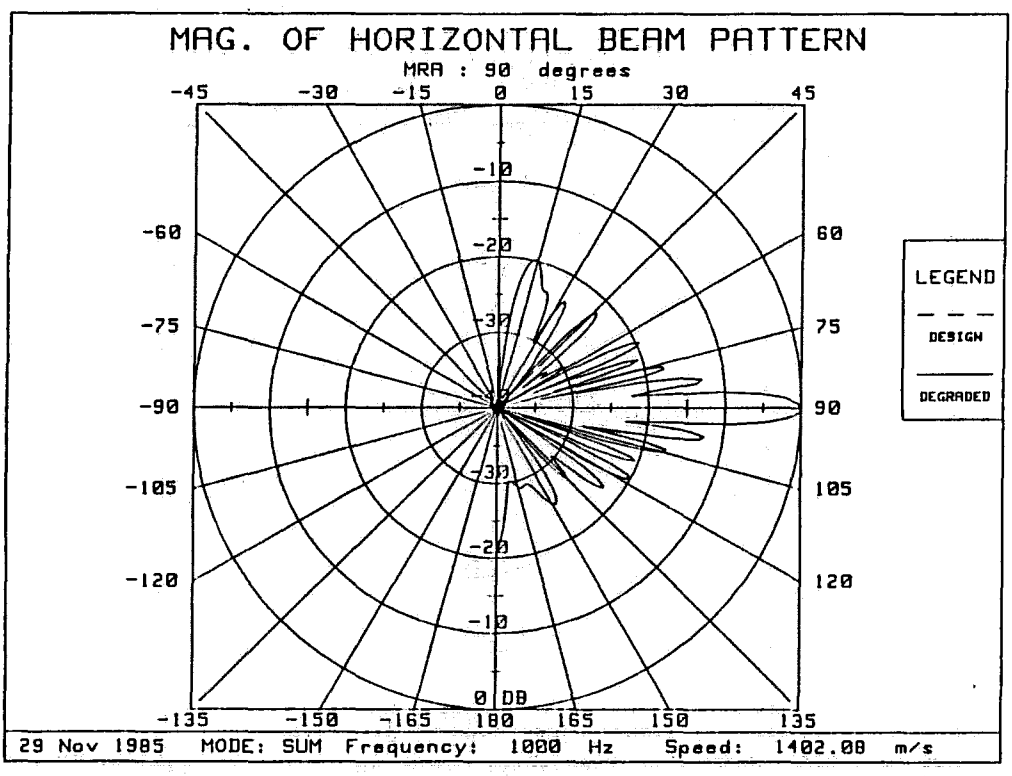

Figure 5.16 Polar Plot of BP at 90 degrees.

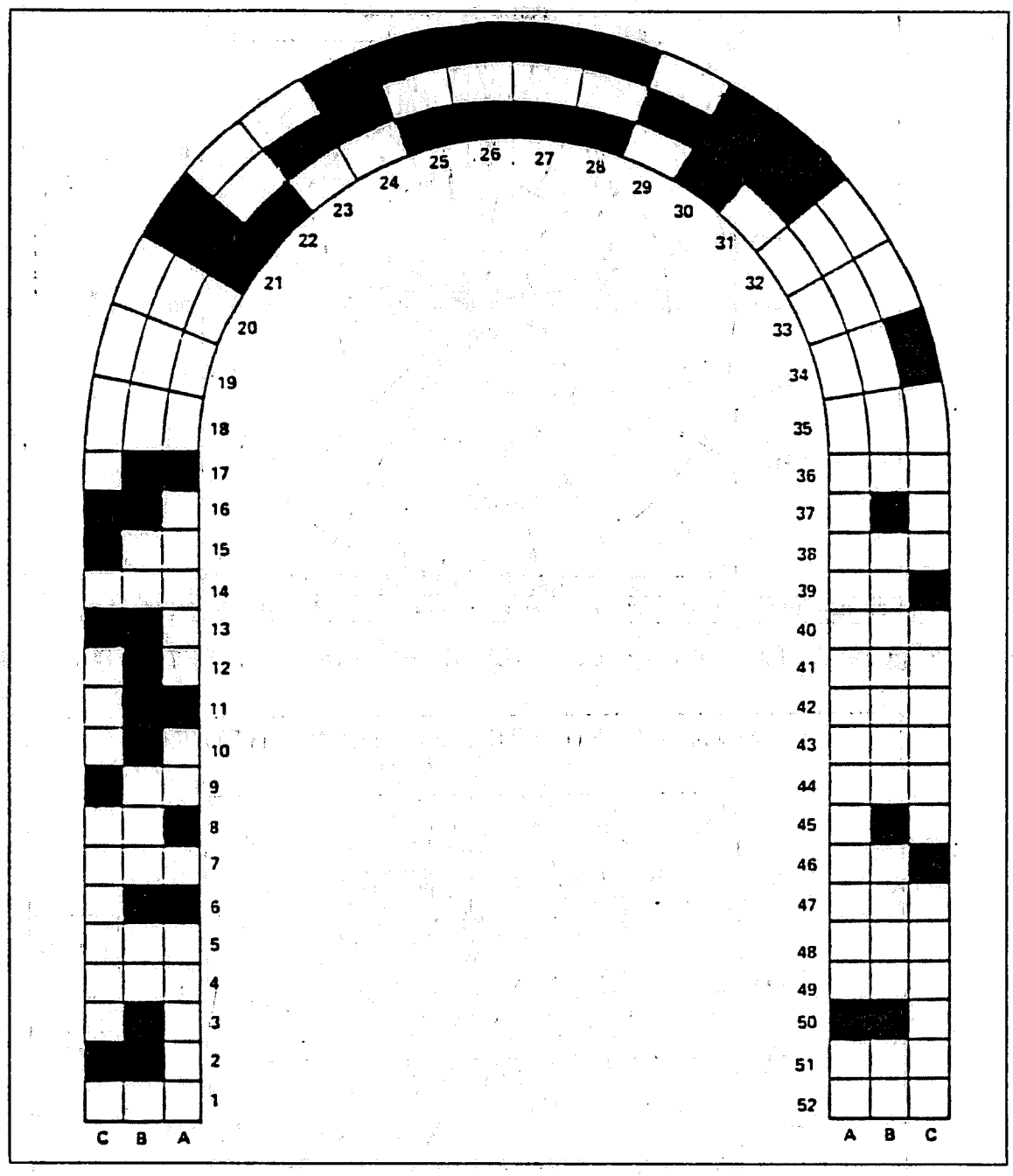

Figure 5.17 Disconnected Hydrophones (in black) for Case 1.

For the "SUM" mode, the sensitivity loss of case one at 1 KHz is about 6 dB and for the case one at 300 Hz, it is about 5 dB. In both cases, the shape of the beam pattern is similar to the theoretical one with an overall loss of

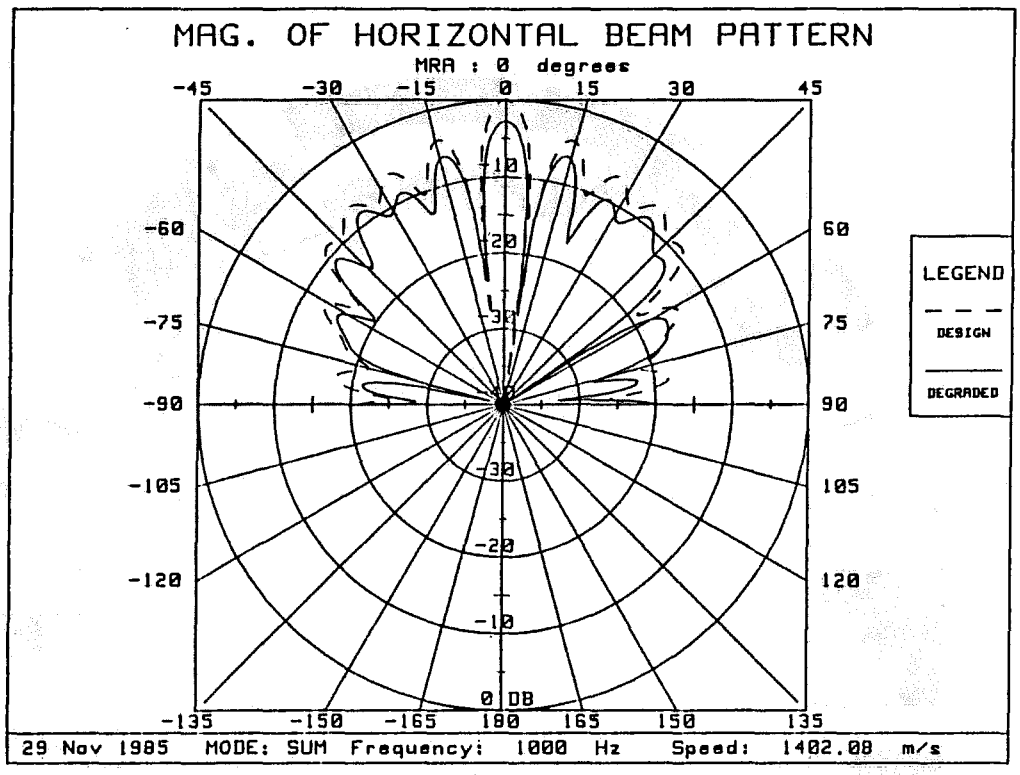

Figure 5.18 Polar Plot of BP - Case 1: 1 KHZ - SUM Mode.

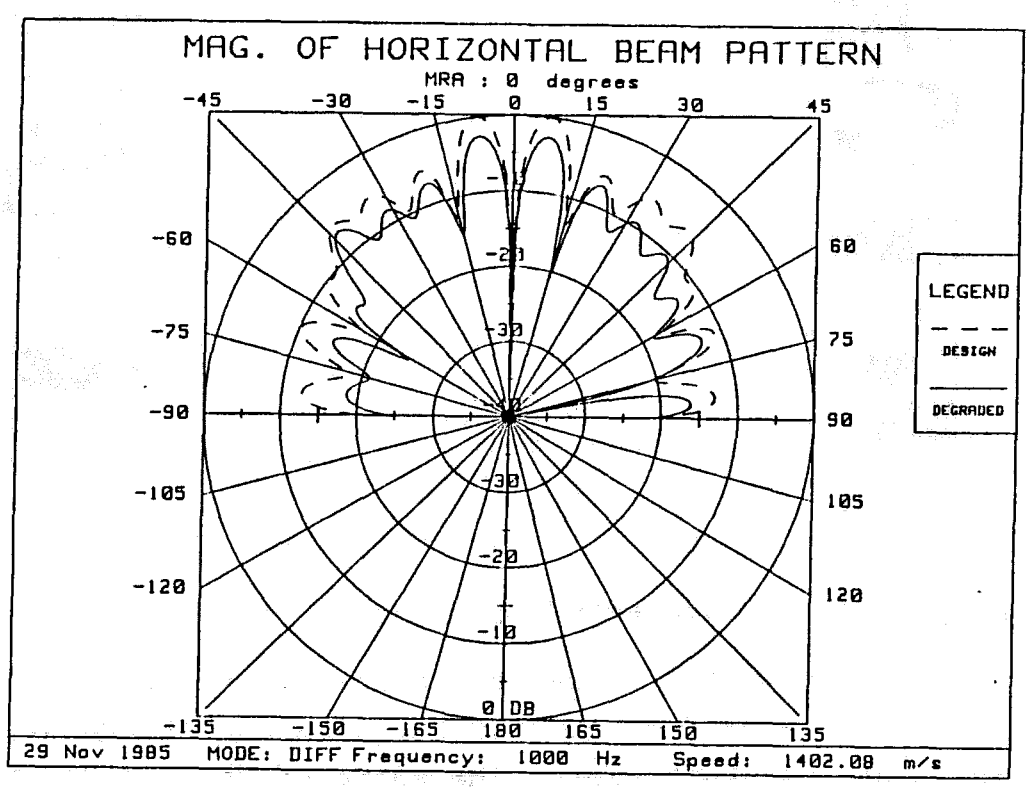

Figure 5.19 Polar Plot of BP - Case 1: 1 KHz - DIFF Mode.

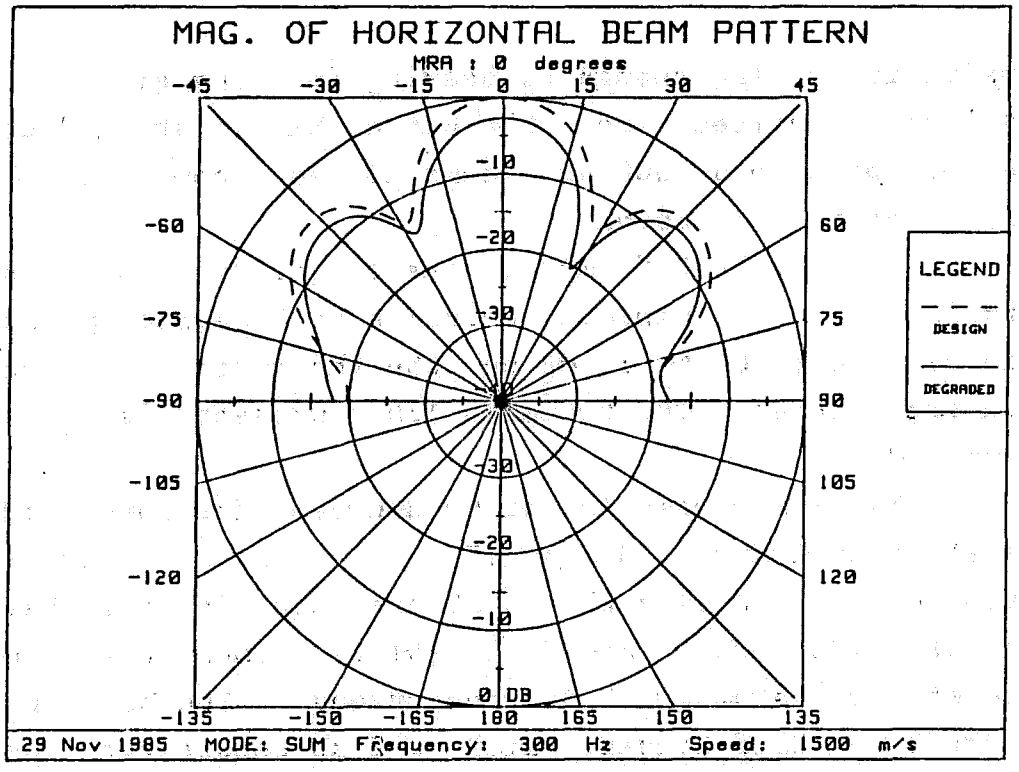

Figure 5.20 Polar Plot of BP - Case 1: 300 Hz - SUM Mode.

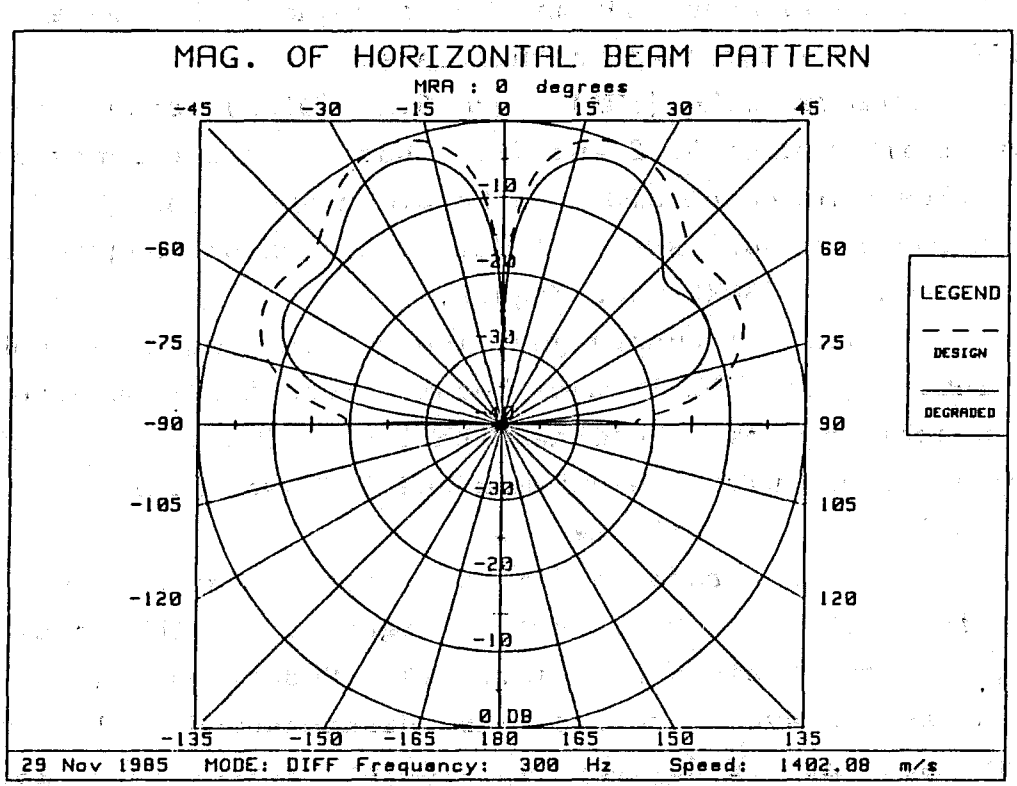

Figure 5.21 Polar Plot of BP - Case 1: 300 Hz - DIFF Mode.

sensitivity. The change of bearing (bearing error) is too small to be noticed, i.e., the actual MRA is still 0 degree. The beam pattern at 300 Hz seems tilted; however, this is only an optical illusion.

For the "DIFF" mode, the sensitivity loss is more significant. The minimum occuring at 0 degree presents a gain of about 10 to 12 dB on the theoretical minimum for both frequencies. The two main lobes present also a loss of about 5 dB.

The next case is at 90 degrees for both modes of operation at 1 KHz. Staves 27, 30, 33, 36, and 39 were disconnected. Staves 27 to 52 are used for this bearing. Figures 5.22 (SUM) and 5.23 (DIFF) show this case of "uneven" or "asymmetric" disconnections. The bearing error is about one-half degree, with a 2 dB sensitivity loss for the "SUM" mode, and also one-half degree of bearing error but 20 dB of gain on the minimum for the "DIFF" mode. The "DIFF" is more affected in this case.

Figures 5.24 (SUM) and 5.25 (DIFF) show the same case as in Figures 5.22 and 5.23 but for a frequency of 300 Hz. The same conclusion can be drawn about the "DIFF" mode as in the previous case. A difference of one degree can be noticed on the "SUM" mode figure.

As the frequency gets lower, and more hydrophones are removed, the "DIFF" mode becomes less and less effective for discriminating a bearing with its minimum. The difference of level with the minimum and its two side lobes also get smaller, up to only few dB.

The last case is for the destruction of an entire section of the conformal array. Figure 5.26 (SUM) shows the beam pattern obtained at 1 KHz, at -90 degrees, where half the array (staves 1 to 13) was disconnected. The total array used at -90 degrees is made of staves 1 to 26. A drastic loss of 8 dB occurs for the degraded main lobe. The

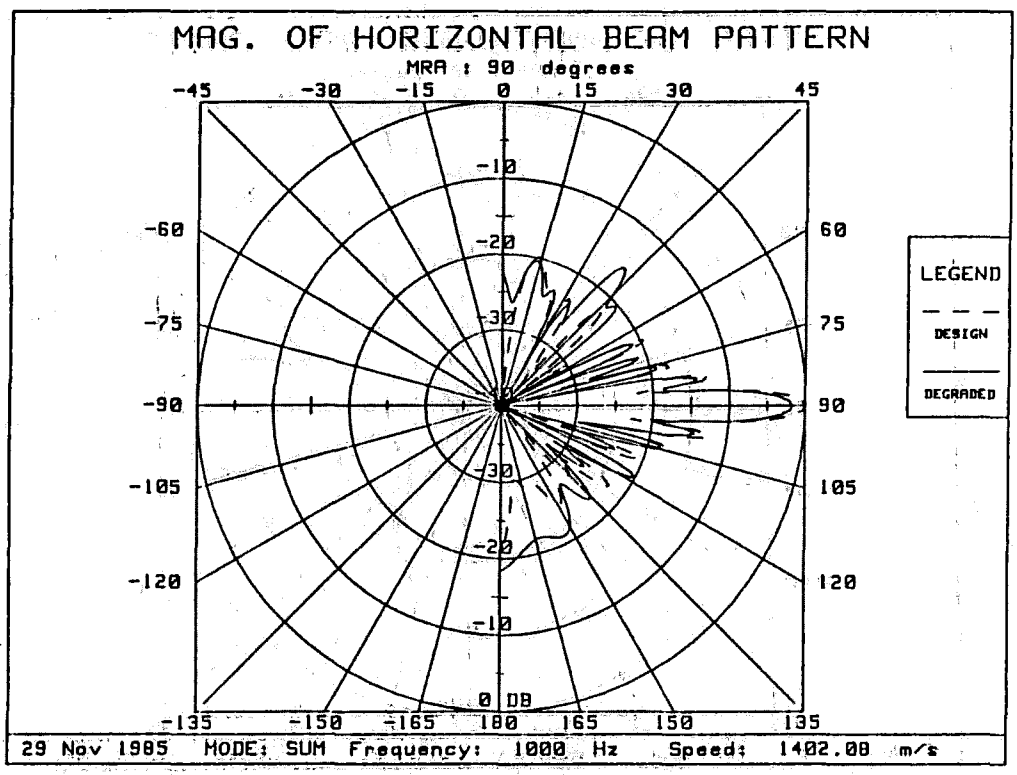

Figure 5.22 Polar Plot of BP - Case 2: 1 KHz - SUM Mode.

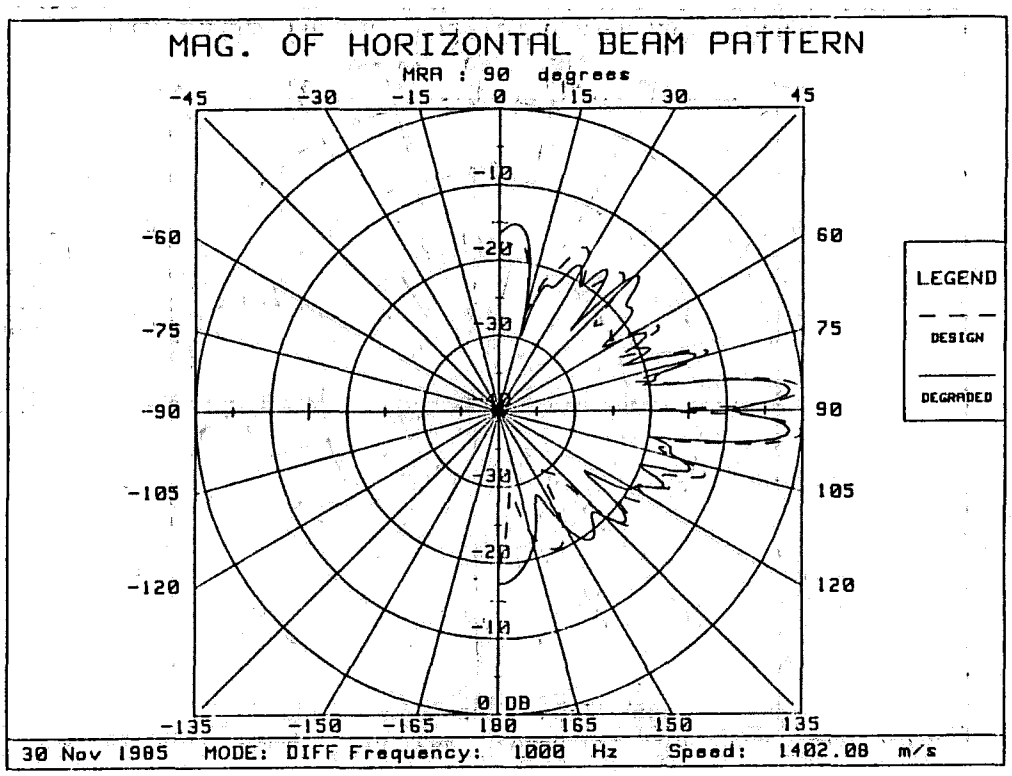

Figure 5.23 Polar Plot of BP - Case 2: 1 KHz - DIFF Mode.

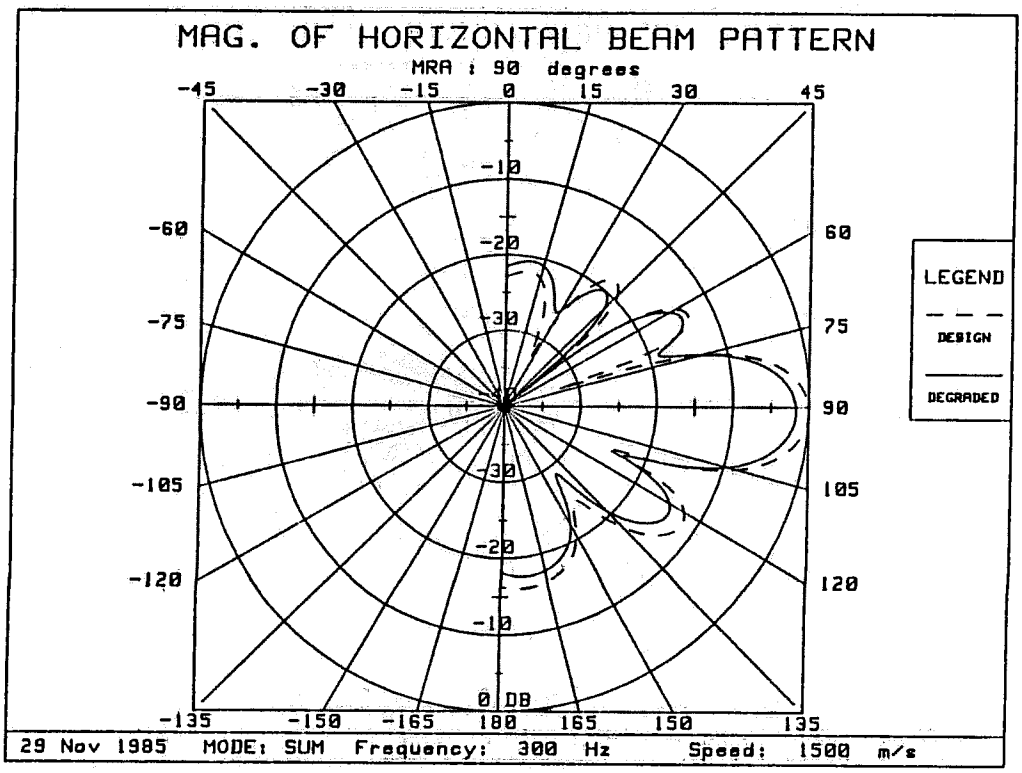

Figure 5.24 Polar Plot of BP - Case 3: 300 Hz - SUM Mode.

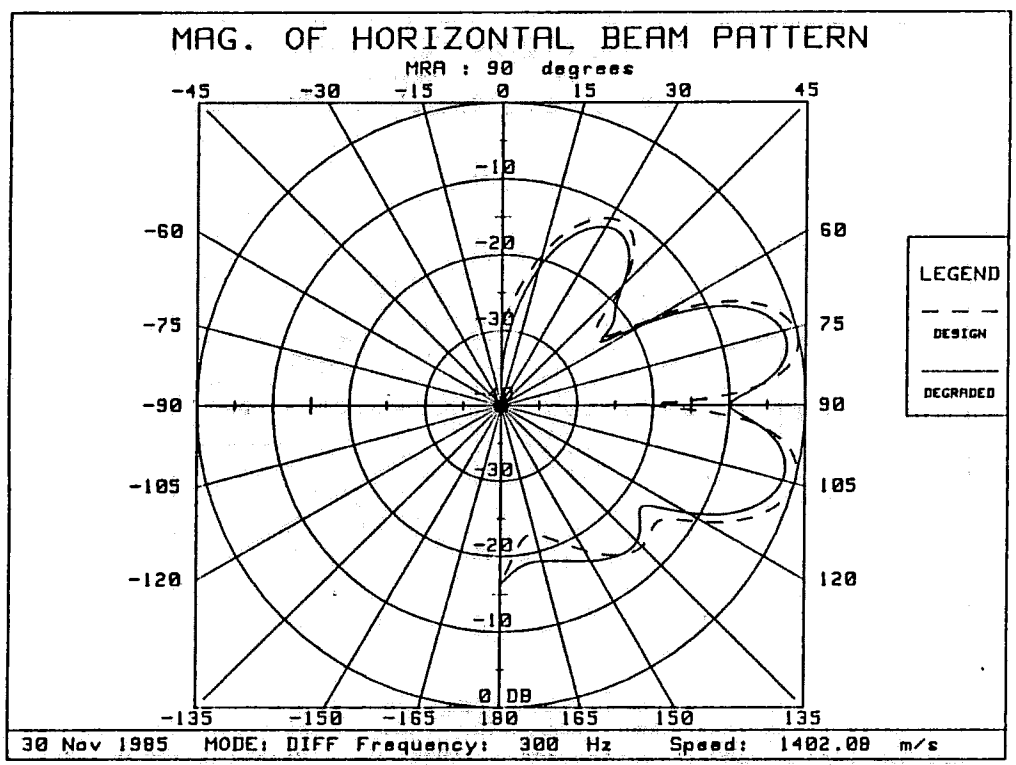

Figure 5.25 Polar Plot of BP - Case 3: 300 Hz - DIFF Mode.

main lobe beamwidth is much larger too. There is no difference between the "DIFF" mode and the "SUM" mode in this case.

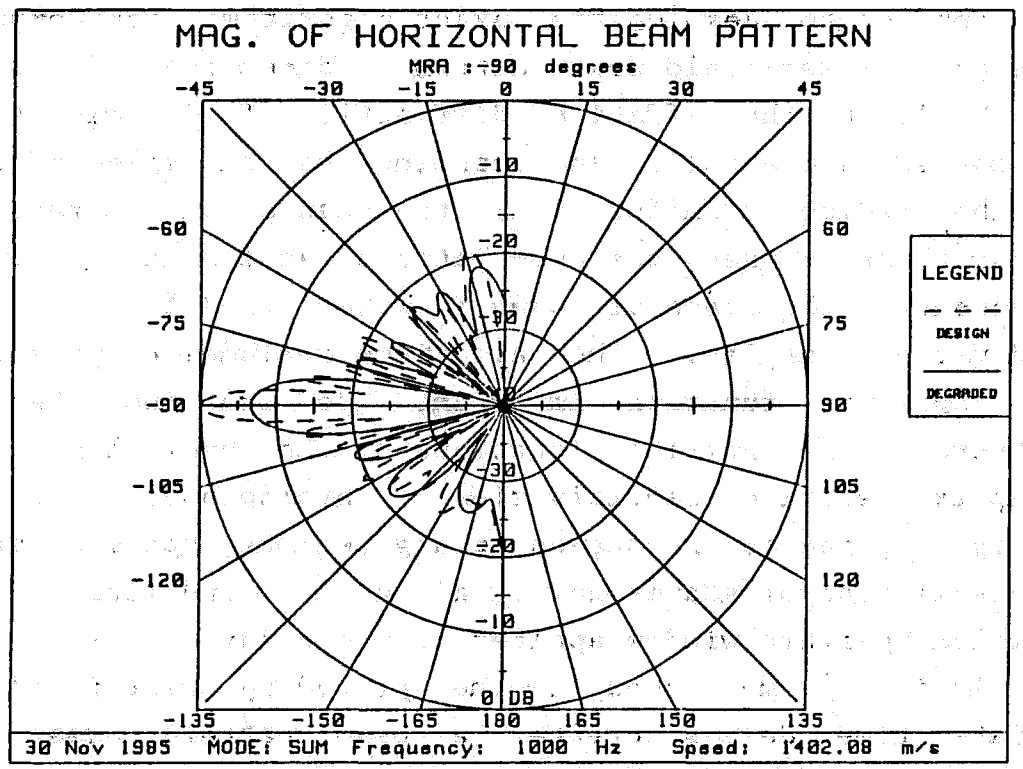

Figure 5.26 Plot of BP with Half an Array Disconnected.

Basically, disconnection affects the sensitivity of the array by 3 to 5 and 8 dB but the bearing errors are relatively small, i.e., one-half to one degree, or simply none. The conformal array was probably well designed for bearing error minimization. The "DIFF" mode is more affected by the asymmetry of disconnection than the "SUM" mode, up to a point were it may become useless.

Finally, even if half the array defined at a given MRA is lost, the bearing error is still small but sensitivity loss is much larger.

B. CONCLUSION

Array theory was successfully applied to a passive, submarine hull-mounted, conformal acoustic array made of hydrophones to develop the exact algorithms for computing the array's far-field beam pattern or directivity function.

Excluding the effect of diffraction of incoming signal by the submarine hull, and assuming that the speed of sound in the medium is uniform over the span of the array, the algorithm expresses the beam pattern as a function of the frequency, the speed of sound in water, and the horizontal and vertical angles. The algorithm includes components of the aperture function important to creating the beam pattern: the physical locations of the hydrophones, the complex relative sensitivities of hydrophone, the time delays applied to the staves at a given MRA, speed of sound, the selection of staves for a given MRA, and finally, the complex aperture window applied to the array.

As it was mentioned in the preceding chapter inapropriate time delays create a beam pattern drastically different, from the one desired.

A stand-alone user-friendly software program running on a micro-computer was successfully designed and developed to simulate the performance of the conformal array as function of frequency, speed of sound, MRA, mode of operation, and the complex relative sensitivities of the hydrophones. It produces the tables of complex sensitivities used for both arrays (theoretical and actual) and of the beam pattern data. It also produces the graphs of the computed beam pattern in both rectangular and polar formats. Moreover, this thesis provides the complete documentation of this program which includes the requirement and design specifications.

Since this is an implementation of a real array, the theoretical array beam pattern should point in the direction

of the MRA, and indeed, the previous section shows that this is the case in the range of frequencies used.

As array theory predicts, when the frequency decreases, the beamwidth of the lobes of the beam pattern increases, while the number of lobes decreases. At the frequency for which the array was designed a narrow main lobe is obtained with small side lobes.

For a given frequency and speed of sound, the beamwidth of the main lobe of the beam pattern is relatively constant for the majority of the MRA. This proves that the original design of the array was adequate. Note: that for a line or a planar array, the beamwidth of the main lobe varies as function of the MRA (bearing), i.e., it is broader at end-fire (0 degree) and narrower at broadside (90 degrees).

The implemented time delays are theoretically valid at its design sound speed. It was found that the beam pattern computed at different sound speed worsen as it differs more and more from the design value. Look-up tables of time delays at different sound speed, such as every 25 m/sec, would greatly enhance the sharpness of the main lobe, while minimizing its side lobes.

The most important effect on the beam pattern is the absence of hydrophones, whether they are disconnected or removed from the conformal array. Some interesting points were found. (Note that the following cases are for the "SUM" mode). At a given MRA, if only one or two hydrophones are missing, the degraded array beam pattern is very close to the theoretical one.

Again at a given MRA, if "symmetrical" degradation occurs, e.g., two staves in the center of the left side and two other staves in the center of the right side of the sub-arrays are missing, the degraded beam pattern still steers in the correct direction (i.e., same as theoretical); however, the degraded array main lobe will be down by 2 to 4 dB from the theoretical one.

If the disconnected hydrophones or staves are at random, or if more are absent on one side of the sub-array, then both the bearing of the main lobe and its magnitude differ from the theoretical ones. However, bearing errors are on the average only 0.5 to 1.0 degree, while the main lobe sensitivity loss may drop by 5 to 8 dB, with a large beam-width in general. The degradation worsens as more and more elements are disconnected at random. It is not possible to predict easily the amount of the bearing error and loss of magnitude for a given case, the program must be run for the specific case.

If very exact bearings are required, one solution is to disconnect some good hydrophones to minimize bearing error while losing some sensitivity. However, this technique may be adequate for a given MRA defining a specific sub-array but it will not work for all MRA's since a new sub-array is defined with different elements for each new MRA; therefore, losing the "symmetry" defined for the other angles.

The best approach is to use an adaptive array. Knowing the actual complex relative sensitivities of every stave, it is possible to electronically compensate for degradation, e.g., by restoring symmetry at a given MRA and frequency. Thus, an optimal array can be defined electronically for any MRA depending of the elements being used. This implementation can consist of applying a variable or adaptive aperture window on the aperture function, in the same way that the "SUM" and the "DIFF" window are used.

Simple cases of better windows were mentioned such as the Hamming and the Dolph-Chebyshev windows. However, the adaptive aperture window should take in consideration the frequency, speed of sound, locations of the hydrophones, sub-arrays defined by the chosen MRA, and of course the actual complex sensitivities of all the hydrophones, and would therefore be more elaborate.

Digital techniques can be used to implement this windowing. It could be incorporated into an updated version of the main program "SUB_ARRAY" to test its feasibility.

For the "DIFF" mode, the absence of few staves does not have a significant effect on the minimum existing at the chosen MRA. The program must be run to visualize the corresponding degradation. However, if more staves are missing on one side of the array compared to the other, the minimum change bearing, and its level will be higher than the theoretical one by easily 10 dB. Furthermore, the "DIFF" mode is much more sensitive to frequency change then is the "SUM" mode. For some frequencies, the minimum is significant, for some others, it is less significant.

The program can be used to try different operational situations, such as the destruction of a whole section of the conformal array. Indeed, it can be used to evaluate whether it is worth while to disconnect or pad a stave for a specific case, etc.

The conformal array was probably well designed since bearing errors are really minimal, i.e., only one-half to one degree of difference with the theoretical prediction is noticeable, even if many staves are disconnected. However, sensitivity of the beam pattern is affected by missing staves, the main lobe level quickly drops by 3 to 5 and 8 dB, along with a generally larger beamwidth.

Finally, this program was made to be adaptable to a large variety of submarine. It is suggested that versions adapted to them be developed.

APPENDIX A

DETAILED DESIGN OF THE PROGRAM SUB_ARRAY

Due to a serious space limitations, it is not possible to provide the detailed design specification of the software program of this thesis. The main program is named "SUB_ARRAY" for submarine array. The detailed design is an extension of the external and architectural design of chapter three. Additionally, the database which is usually provided with the detailed design is available in Appendix B.

A. CHARTS

This appendix will provide, firstly, some more details about the main modules of the design section for both data flow diagrams and structured charts, and secondly, two pseudocode lists of two routines as an illustration.

The first module is "Compute the normalized beam pattern" that is shown in Figure 3.10 Again, the entering arrows indicate the input, the exiting ones indicate the output. The logic applies for both theoreitical and actual array beam pattern, except that the beam patterns are norromalized using one common normalization factor, so both beam patterns can be compared for degradation.

The required aperture function containing the staves selection (sub-arrays), number of elements used, their locations, their complex weights, their time delays, the frequency, and the wavelength, is used to "Compute the real and imaginary parts of the beam pattern by DFT". A data structure (here an array) is filled by the horizontal angles used for the calculation.

The polar representation (magnitude and phase), of the beam pattern is computed; however, only its magnitude is

used. The normalization is also computed and then applied to the magnitude of the beam pattern. The normalization factor is the highest value of the lobes of both beam patterns.

The normalized magnitude of the beam pattern is used to compute its logarithmic representation, i.e., the magnitude in decibels. Finally, for the purpose of doing the polar plots, the beam pattern components for the polar plots are computed in decibels.

The following outputs are available for both theoretical and actual array cases:

1. Normalized magnitude of the beam pattern;
2. Normalized magnitude of the beam pattern in dB;
3. Polar plots components of the beam pattern in dB; and
4. Horizontal angles.

The next module is the "Configure array" as shown in Figure 3.11 First, the requested MRA of the user is change to the MRA used for the selection of the staves of a sub-array, eg., a request for 92.6 degrees is rounded off to 95 degrees. Then the staves are selected. The complex weights are defined, and the spatial locations of each hydrophones are identified. The final result is a preliminary aperture function.

The menu "Choose graph output medium" allows the user to choose the medium for each graph type requested. Figure 3.13 shows the corresponding data flow diagram. Basically, if no graphs are requested a message is displayed for five seconds. If any graph is demanded, the plot is done on the CRT (screen), then a menu of medium and size options is presented to the user. His request is then plotted, or printed, or displayed on the requested medium.

The last menu, called "Modify Individual stave status", is more elaborated. At many points, the user can exit for this menu if no changes are required. User choices for

changes are: disconnect a stave or any of its hydrophones, pad any stave with a number of capacitors which is function of frequency, and finally change the complex weights of the hydrophones of a selected stave. Each stave entry is validated against the selected MRA, i.e., stave of a defined sub-array. When changing complex weights, all entries are validated. Moreover, the old and new complex weights of the hydrophones are shown to the user. Padding consists of adding up to three capacitors in parallel to the selected stave, according to the user choices. The effects of padding are predefined by the program. Figure 3.14 shows the detailed data flow diagram of this menu.

Using the main data flow diagram, the structured chart of the program is then made. A few examples are provided in order to assist in understanding a structured chart. Using Figure 3.15, "User menu", allows the user to change the parameters of the program if the standard one is not desired. The parameter is selected, the request control flag for the option is passed to the submodule "Enter changes", the user enters the change which is passed to the sub-module "Validate changes" for validation. The new value (change) is then passed to the sub-module "Perform change of parameter value" for incorporation of the new value into the database. The user does not have direct access to the database since at least three sub-modules are required to properly perform any changes. The parameters are then passed to the main program Sub-array to be used by other modules through the database.

The module, "Configure array as an aperture function" of Figure 3.15, defines the MRA index to be used to gain access to the data stores using the requested MRA. The parameters or values of the MRA index are then passed to three sub-modules which the configure the aperture function which is itself passed back to the main program. The three sub-

modules under "define MRA index to use" could be done concurrently, if the computer were capable of doing it. It cannot be done with the HP 200 series computer.

Another example, the module "Individual stave change (Menu)" of Figure 3.15 shows only the principal features. Basically, a stave is choosen, then a selection is made of any options, the options and stave number and any relevant data and control flags are passed to the next three sub-modules. Only the selected sub-module is then executed. The control is then passed back up to the entry point, i.e., the menu. Indeed, all the module are like that, the modules are hierarchically organized, such that a sub-module is accessed through the module or sub-modules "above" it. Thus, the sub-module "Choose medium (Menu)" is accessed through the "Output graphs" module.

Figure 3.17 shows the ten selections available to the user with the user menu which allows the change of specific parameters. The choices are those specified in the software requirements, plus a selection to request the execution of the program when all the entries are made (if any). Softkeys are used for selection.

Figure 3.18 shows the structured chart of the next menu which allows the change of individual stave status. Softkeys are used for selection. At least five sub-menus are identifiable:

1. User needs of the menu - i.e., change required or not;
2. Select options - Disconnect, Pad, Change individual staves, or Exit;
3. Disconnect a stave or any hydrophone - select the hydrophone number;
4. Select the number of capacitors - 1, 2 or 3; and
5. Change the individual hydrophone complex weights (select amplitude and/or phase).

Again validation of all entry is made when they affect the database, in this case the complex weights.

Figures 3.20 and 3.21 show the details of the modules which output the tables of the aperture function and the beam pattern data (or function). The modules are executed even if the table is not requested. The sub-module "Check if requested" which will pass the appropriate control flag for execution to the other modules. In both modules, the user has to specify whether the table is to be printed or displayed on the CRT.

Finally, Figure 3.22 presents the "Output graph" module which controls the output of all graphs along with the menu for the selection of size of graphs and medium for output. The user can choose the CRT, the plotter, or the printer (Thinkjet or thermal printer) with the address specified in the requirements. Two sizes of graphs are available from the printer, normal size (about 4 inches high and 5-1/2 inches wide) and expanded (oriented sideways covering the entire page). However, the quality of the expanded view is not very good, and it should be used only to visualize the curves better. Again three graph types are available.

B. PSEUDOCODE

The first example is for the "User_menu" module.

```
USER_MENU:
   Clear the screen;
   Output choice of parameters with default values;
   Activate softkeys;
   Define softkeys with actions to be taken when chosen;
     For key 0 to key 9;

   WAIT FOR softkey interrupt, GOTO proper key action;

 KEY 0:  (Output hydrophones sensitivity)
   IF output in data flag is TRUE THEN FALSE
   ELSE TRUE;
```

```
KEY 1:  (Output beam pattern data)
  IF output out data flag is TRUE THEN FALSE
  ELSE TRUE;

KEY 2:  (Graph of beam pattern in dB)
  IF graph in dB flag is TRUE THEN FALSE
  ELSE TRUE;

KEY 3:  (Graph of magnitude of beam pattern)
  IF graph magn. of BP is TRUE THEN FALSE
  ELSE FALSE;

KEY 4:  (Polar plot of beam pattern)
  IF graph polar of BP is TRUE THEN FALSE
  ELSE TRUE;

KEY 5:  (Enter the frequency)
  REPEAT
       Enter data from operator
       Test if positive number
       IF TRUE THEN
           Convert into a number
           Define frequency
       ELSE
           Display a message about the user error
       END IF
       IF frequency = 0 THEN
           Inform user about his error
       END IF
  UNTIL frequency is valid;
  Wavelength = Speed/Frequency;

KEY 6:  (Enter the MRA)
  REPEAT
       Enter data from operator
       Test if it is a number
       IF TRUE THEN
           Convert into a number
```

```
                IF number > 180 AND < -180 THEN
                     Inform user of invalid MRA
                ELSE Define the MRA
            ELSE
                Display a message about the user error
            END IF
        UNTIL MRA is valid;

    KEY 7:  (Enter the speed of sound)
        REPEAT
            Enter data from operator
            Test if it is a positive number
            IF TRUE THEN
                Convert into a number
                Define MRA
                IF number > 1600 AND < 1400 THEN
                     Inform user of invalid sound speed
            ELSE
                Display a message about the user error
            END IF
        UNTIL sound speed is valid;
        Wavelength = Speed/frequency;

    KEY 9:  (Define the mode of operation)
        IF Mode of operation = SUM THEN DIFF
        ELSE SUM;

     Clean up the screen;
     Deactivate softkeys;
End of User_menu
```

The second example is for the function which tests whether a passed line of data variable contains a number, and it returns "TRUE" if it is the case.

```
FUNCTION Is_string_num (Data_line);
     IF Data_line = NULL OR empty THEN
```

```
        RETURN FALSE
    ELSE
        Trim all leading zero from Data_line;
        Test first character
        IF character NOT = "-" THEN
            Safe that character
        ELSE
            Skip the "-";
            Take the next charater
        END IF;
        IF (character > "9") AND (character < "0") THEN
          RETURN FALSE  (It is not a number)
        ELSE RETURN TRUE;  (It is a number)
    END IF (OF Data_line);
RETURN The value
```

APPENDIX B

DATABASE OF THE SOFTWARE PROGRAM SUB_ARRAY

A. THE DATABASE

HP BASIC has limited data structure types available; however, they will be all used. Only the main elements of the database will be presented here. This appendix shows the entire data base of the program. It is fully documented.

The formal way to design the data base and all the variables can be extremly lengthy and usually covers as many pages as there are variables (about one hundred). Therefore, an informal approach will be used.

All control flags are strings which simulate boolean variable (TRUE/FALSE). Instead of TRUE/FALSE, there are some cases where the word "YES"/"NO" are used when it is more meaningful to do so.

The variable "Max_nn_staves" (integer) indicates the maximum number of staves in the entire conformal array which is 52 presently. It must have a value of two at least. The maximum available integer number, is about 32,000.

The amplitude and phase weights of the theoretical and actual arrays are stored in four two-dimensionnal real arrays of 1 to 52 (or Max_nn_stave) for the staves and of -1 to 1 for three hydrophones of a stave. The magnitudes are normalized from 0 to 1.0 (a linear scale), and the phases are in radians. The variables are:

1. Actual_amp_wg (1:52,-1:1);
2. Actual_pha_wg (1:52,-1,1);
3. Theory_amp_wg (1:52,-1:1); and
4. Theory_pha_wg (1:52,-1:1).

The time delays are in seconds. They are stored in an array of real element of 1 to 52. The name of the variable is "Time_delay".

All spatial locations are stored in three real arrays. X and Y are positions of the staves (1 to 52). Z is a -1 to 1 real array for the vertical position of each hydrophone, the same distance for every stave. The names are

1. X_psn (1:52);
2. Y_psn (1:52); and
3. Z_psn (-1:1).

If another implementation requires a new Z positions for every elements, the variable can become a (1:52)(1:52)(-1:1) array. Appendix B shows the details.

The next structures deal with the different format for the beam patterns for both theoretical and actual arrays. Two real arrays of -180 to +180 for the real and the imaginary parts of the beam pattern are defined. Similar arrays for the magnitude and phase, and for the dB representation are defined. The index of the arrays is from -180 to +180, i.e., one data value for every degree of the horizontal angle, since all beam patterns are function of the horizontal angles. Additionally, two more arrays are required to express the magnitude of the beam pattern in decibels into the x and y axes of a polar plot. This is done to simplify the drawing of the curves using the HPGL which requires x, y pair of points instead of radius and angle.

The names of the variable are:

1. Theory_re_bp (-180:180) for real part;
2. Theory_im_bp (-180:180) for the imaginary part;
3. Theory_mag_bp (-180:180) for the magnitude;
4. Theory_phase_bp (-180:180) for the phase;
5. Theory_db_bp (-180:180) for the magnitude in dB;
6. Theory_db_xaxis (-180:180) for the x axis of polar plot; and

7. Theory_db_yaxis (-180:180) for the y axis of polar plots.

All "Actual" array terms are the same as for "theory", except that the word "Actual" is used instead of "theory".

If an implementation requires the use of both vertical and horizontal angles, it is a simple matter to add another dimension of -180 to 180, or 0 to 360, or whatever angular range is required. As an example, if one data point is required per degree, the real part becomes Actual_re_bp (-180:180,0:360) the first index is for the horizontal angle, the second for the vertical one. If only the vertical angles -10 to 10 degrees are required, then the variable becomes Actual_re_bp (-180:180,-10:10). Of course, if more data point are required, e.g., every one-tenth of a degree, the variable is changed to (-1800:1800), since the index is not used for the angle as such.

The value of the horizontal angle is stored in a real array of -180 to 180 elements in degrees named Psi. As mentionned before, if more data points are required, the size of the array is adjusted accordingly, and the values of the array are defined. A variable called Theta can also be defined for the vertical angle if it is required.

Two buffers are used to read the time delays from the data stores of time delays. One is called "time_del_buffer(1:28)" (real) for transfering the read data onto the array used for beam pattern calculation, i.e., "Time_delay". The second one "Temp_td_buffer" is used to convert the time delays defined for a positive MRA to its mirror image for a negative MRA. Both buffers are used by the routine "Define_time_del". The dimension 1 to 28 is defined by the maximum number of elements in any sub-arrays (for a given MRA).

The next set of variables is for the MRA look-up table. The real array "Mra" from -38 to +38 is an array of angles

in degrees from -170 to 170 degrees defining the angle at which a set of sub-arrays is defined, eg., at 0, 5, 10, 15, ..., 160, 165, 166, etc. Thus, if the user needs to use an MRA of 6 degrees, the sub-arrays defined by MRA at 5 degrees will be used. The index -38 to +38 is defined by the variable "Mra_index" (integer) which points to the information defined by the look-up table. As an example, if the Mra_index is equal to 1, then Mra(Mra_index) is equal to 5 degrees, etc. The identification of the index for a requested MRA, is done by the routine "Find_mra_index".

Up to four sub-arrays can be defined at a given MRA, the first and second left sub-arrays, and the first and second right sub-arrays. There are always a first left and a first right sub-arrays. In this particular system, there is no second left sub-array, and for a few angles a second right sub-array is defined. All sub-arrays are contiguous. Therefore, the first and last elements are required to define a sub-array.

These four sub-arrays are two-dimensional from -38 to 38 for the MRA index and 1 to 2 for the number of the stave, i.e., index 1 is for the first element of the sub-array, and index 2 is for the last element. The names are:

1. Left_1st_array (-38:38,1:2);
2. Left_2nd_array (-38:38,1:2);
3. Right_1st_array (-38:38,1:2); and
4. Right_2nd_array (-38:38,1:2);

The values stored in these arrays vary from 0 to 52 where 1 to 52 indicates the stave number, and 0 is used to indicate no stave number. The value 0 is necessary since an array may consist of only one or no element. The index identified by the value 1 indicates the number of the unique stave of this sub-array, and the second index with 2 has a value of 0, or if no elements are used then both are zero.

To accelerate the identification process of all second sub-arrays, two flags are used to indicate whether there is a second sub-array or not. They are:

1. Flag_2nd_left$(-38:38){1}; and
2. Flag_2nd_right$(-38:38){1}.

These arrays of one character string are defined for every Mra_index, where a flag "Y" indicates the existence of the second sub-array, and "N" indicates that there is no second sub-arrays. One is used for the left side, the other for the right side.

Currently, the section of code where the computation for the left second array is done is deactivated by an additional flag to accelerate the computation. The flag is either "TRUE" (activated) or "FALSE" (deactivated), and it is defined as "Left_activate${5}".

An array of integers stores the total number of elements defined at a given MRA by the Mra_index, another one stores the total number of elements in the left sub-array (both first and second), and similarly for the right sub-array. They are:

1. Tln_nn_element (-38:38);
2. Nn_left_element (-38:38); and
3. Nn_rght_element (-38:38).

For the menu "Individual stave change", two important flags are used. One indicates if the user wants to use the menu, i.e., to do any changes ("TRUE"), or not ("FALSE"). It is defined as "Stave_menu_flag${5}". The second one is to indicate whether more staves have to be changed or not after the user has accessed the menu. It is defined as "Stay_in_menu${5}".

The main menu (user menu) and the main routine for graph output use some important flags for controlling which graph is to be made, where, and with what size. The three first flags indicate whether the user wants a certain graph ("YES") or not ("NO"), and they are:

1. Graph_ma_flag${3} for the graph of normalized magnitude;
2. Graph_db_flag${3} for the graph of the magnitude in dB; and
3. Graph_po_flag${3} for the polar plot in dB.

Of course, all flag can be "YES" or all "NO" (no graph at all). The next one called "Graph_op_medium${4}" indicates the type of output medium used for the graph and its size (expanded or normal). Presently, six choices are available, and they are:

1. CRT = CRT output;
2. PRHN = Printer HP2671G (Thermal printer) normal size;
3. PRTN = Printer Thinkjet normal size;
4. PRHN = Printer HP2671G (Thermal printer) expanded size;
5. PRTN = Printer Thinkjet expanded size; and
6. PLOT = Plotter normal size (full page in this case).

When "PLOT" is chosen, a flag called "Do_plot${5}" is set to "TRUE" to indicate the request to the graph routine.

For the output tables, two flags indicate the choice of the user, i.e., "YES" or "NO". One is for the output of the aperture function, the other for the output of the beam pattern (logarithmic or linear). The three character string flags are:

1. Output_in_data${3} for the aperture function; and
2. Output_bp_data${3} for the beam pattern data (theoretical and actual).

Routines "Display_input" and "Display_output" also use a flag to indicate whether the data are output on the line printer or not ("Y" or "N"). The name of the flag is "Lp_input_flag${1}" where Lp means line printer.

For the menu "individual staves change", some integer variables are used. The first, called "Stave_nn_select", indicates the current choice of stave to be changed. It

varies between 1 and 52 but is validated against the choice of staves defined at a given MRA. The other indicates the choice of hydrophone number to be changed for the selected stave, and is called "Hydro_nn". It varies from -1 to +1. For padding, the number of capacitors is specified by the variable "Padding_level" from zero to three, where zero means no capacitors, and three means three capacitors.

For the MRA index, the user chooses an MRA between -180 to 180 degrees, which can be 93.689 degrees for example. This value is stored in "Requested_mra" (real). It is then adjusted for the system program in the form of a rounded MRA, eg., 95 degrees, in accordance with the MRA choices defined in the data store (the array Mra). The name of the variable is "Used_mra" (real).

Finally, standard data variables are defined for some of the physical parameters of the program. These real values are:

1. Freq for frequency in Hertz;
2. Speed for the speed of sound in the medium in m/sec;
3. Lambda is for the wavelength in m;
4. Normal_factor is for the normalization factor of the beam pattern; and
5. Temp_mag and Temp_phase are used to temporarily store the values of magnitude and phase of the relative complex sensitivity of an hydrophone or a stave.

B. LOOK-UP TABLES AND DATA STORES

Four data stores are used in the program, three are from the internal data store (with DATA statement), one is external from disk. Eventually, a fifth data store could be defined for the aperture window coefficients if more elaborate window algorithms are used such as an Hamming, or a Blackman, or a Dolph-Chebyshev window.

The first data store is for the definition of the X, Y, and Z spatial positions of all the staves and hydrophones of the conformal array.

In routine "Define_hydr_psn", DATA statements constitute the look-up table for the x and y positions in meters. They are read and assigned to the X and Y positions of all staves from 1 to 26. Then a mirror image is made for the other 26 staves. Tthe values for the x and y positions are adjusted in accordance with the coordinate system of Chapter 2, i.e., the origin is in the middle of the conformal array. The Z position is only defined by assignment statements for the three hydrophones of a stave. Table VI provides the resulting x and y position in meters for all the staves.

The values for the Z position are:

1. Hydrophone A: -0.762 m;
2. Hydrophone B: 0.0 m; and
3. Hydrophone C: +0.762 m.

The data are read then transferred into the position array "X_psn", "Y_psn", and "Z_psn".

The next data store is for the information relevant to the sub-arrays defined by a specific array. The variable mentioned in the section about the database are assigned the values defined by the DATA statements (forming a look-up table) in the routine "Array_defn". The MRA varies from 0 to 165 degrees in steps of 5 degrees, then from 166 to 170 degrees in steps of 1 degree. The negative counterpart of MRA is a mirror image of the positive MRA, e.g., stave 1 correspond to stave 52, etc.

For a given MRA, a DATA statement contains the following values:

1. MRA value, e.g., 165 (165 degrees);
2. Total number of elements, e.g., 17;
3. Number of elements in the left sub-arrays, e.g., 9;
4. Flag of left second sub-array, e.g. "N";
5. First element of first left sub-array, e.g., 37;
6. Second element of first left sub-array, e.g., 45 (i.e. from 37 to 45);

TABLE VI

X AND Y POSITIONS OF THE STAVES

STAVE ##	X POSITION meters	Y POSITION meters
1	-7.323	-4.382
2	-6.719	-4.331
3	-6.115	-4.277
4	-5.589	-4.254
5	-4.909	-4.158
6	-4.306	-4.093
7	-3.704	-4.020
8	-3.194	-3.963
9	-2.530	-3.866
10	-1.899	-3.780
11	-1.301	-3.689
12	-.700	-3.594
13	-.104	-3.489
14	.492	-3.380
15	.980	-3.265
16	1.797	-3.072
17	2.263	-2.970
18	2.846	-2.805
19	3.424	-2.619
20	3.991	-2.408
21	4.549	-2.172
22	5.127	-1.911
23	5.608	-1.608
24	6.080	-1.226
25	6.535	-.818
26	6.859	-.305
27	6.859	.305
28	6.535	.818
29	6.080	1.226
30	5.608	1.608
31	5.127	1.911
32	4.549	2.172
33	3.991	2.408
34	3.424	2.619
35	2.846	2.805
36	2.263	2.970
37	1.797	3.072
38	.980	3.265
39	.492	3.380
40	-.104	3.489
41	-.700	3.594
42	-1.301	3.689
43	-1.899	3.780
44	-2.530	3.866
45	-3.194	3.963
46	-3.704	4.020
47	-4.306	4.093
48	-4.909	4.158
49	-5.589	4.254
50	-6.115	4.277
51	-6.719	4.331
52	-7.323	4.382

7. First element of the second left sub-array, e.g., 0;
8. Second element of the second left sub-array, e.g., 0;
9. Number of elements in the right sub-array, e.g. 8;

10. Flag of right second sub-array, e.g. "Y";
11. First element of first right sub-array, e.g., 46;
12. Second element of first right sub-array, e.g., 52 (i.e. from 46 to 52);
13. First element of the second right sub-array, e.g., 1 (only one element);
14. Second element of the second left sub-array, e.g., 0;

Table I shows the complete stave selection for any positive MRA implemented for this program.

The next data store is for the time delays applied to a stave/hydrophones at a given MRA and for a given speed of sound. Only few real time delays were implemented, the others receive a linear time delays of 61.5 microseconds starting from the center of the array and increasing outward (i.e., 61.5, 123.0, 184.5, etc.). The left side sub-array receives negative time delays, the right one a positive delays.

For the real time delays presently available, the routine "Define_time_del" has the values of the time delays expressed as DATA statements forming a look-up table. For the MRA with real time delays, the data are read and put into a buffer of 28 elements. If fewer time delays are required for a sub-array with fewer staves, the remainder of the 28 elements are assigned a delays of 0. They are then assigned to their proper location in the array "Time_delay" to be used for the calculation of the beam pattern. The delays are in seconds and are positive or negative.

For examples, for +90 degrees, the first datum is about -2500 microseconds which is then assigned to the 27th term (i.e. stave) of the array "Time_delay". The 26th data read is the last one needed for the sub-arrays, since only 26 staves are used at 90 degrees, and it is about +2500 microseconds. It is assigned to the 52nd term of the array "Time_delay". Since the buffer contains 28 terms, the last two remaining terms are equal to zero.

For negative MRA, the time delays are also the mirror image of the positive MRA time delays. The "Time_del_buffer" is also used to skip the time delays of other MRA, e.g., skip delays of 0, 5, 10, ..., up to 85 for the MRA of 90 degrees. This is required since the data statements cannot be randomly accessed.

The last data store is on disk. For every frequency group defined in the program, such as 50, 100, 150, 200, 300, etc. up to 5000 Hz, a file exists on a disk containing the relative complex sensitivities of all 52 staves. This file is generated by a different program which produce files of magnitude in decibels and phase in degrees for all 52 staves at a given frequency from 52 files of complex sensitivities as function of frequency. The values are converted in a linear scale and in radians into the data variables for the actual array (see database section). The theoretical complex sensitivities are defined by direct assignments of 1.0 for the magnitude and 0 for the phase.

If no disks are used, the actual complex sensitivities are defined as the theoretical values, and the user can then change the values with the "Individual staves change" menu.

C. DATABASE DEFINITION LISTING

The database is defined at lines 1450 to 2800 of the main listing in Appendix E.

APPENDIX C
USERS MANUAL

A. INTRODUCTION

1. Product Overview

The software program "SUB_ARRAY" for submarine array has been developed for the purpose of predicting the extent of array performance degradation resulting from missing, padded or complex weighted hydrophones in the AN/BQR-7 passive sonar system. The program is a stand-alone user-friendly package running on any HP 9000 Series 200 micro-computer with extended memory and floating point co-processor. The program is written in HP BASIC 3.0 language (a structured programming language) and is fully documented.

For given submarine type, the program simulates the performance of the conformal array as function of frequency, speed of sound, maximum response angle or MRA (bearing), mode of operation, and the actual complex relative sensitivity of the hydrophones.

The algorithms used are based on array theory. Both theoretical and degraded (actual) far-field beam patterns or directivity functions are computed. The program produces both tabular and graphical representations of the output and the complex sensitivities used. Sensitivity loss and tracking correction are also easily readable from the graphs.

Many self-explanatory menus are used to reduce to simplify the process of putting in data. All input parameters have default values.

2. Terminology and Basic Features

The far-field beam pattern is also called the directivity function. It is the angular functional dependence of the normalized sound pressure amplitude at a large distance from the source. The maximum response angle (MRA) is the bearing of the main lobe in degrees. The MRA varies from -180 to 180 degrees, where 0 degrees is the straight ahead bearing towardthe bow of the submarine. The hydrophone's complex relative sensitivity, also called complex weight, is basically the expression of the frequency response of the hydrophone. It is expressed as a magnitude in dB from 0 dB to -40 dB and a phase from 0 to 360 degrees. A stave is a group of three hydrophones connected in parallel and aligned vertically. These three hydrophones are labeled A, B, and C or -1, 0, and 1.

The program has three main menus. The first one allows the operator to input the basic parameters of the program, i.e., frequency (Hz), speed of sound (m/sec), MRA (degrees), and mode of operation ("SUM" or "DIFF"), and also the types of output, i.e., table of complex sensitivities, table of beam pattern data, the three graph types of the beam pattern.

The second menu allows the operator to change the status of the individual hydrophones used at a given MRA. All changes are temporary for a run, it is not stored on disk, nor on a permanent tables. When the program is executed again, the changes are lost. The following three actions can be taken with this second menu, and they are all controlled by a sub-menu:

1. Disconnect a stave or any of its hydrophones;
2. Pad a stave with one, two, or three capacitors; and
3. Change the complex sensitivities (magnitude and phase) of any hydrophones.

After the beam patterns had been computed, which requires about two to four minutes, a last menu is used to allow the operator to make a choice of the size of the graph (expanded or normal for the printer) and the medium (screen, printer, or plotter).

The program goes on a "PAUSE" mode while the user analyse a graph, then the next graph, if requested, will be displayed at the user command.

The user is kept informed at all times of what is being done. All inputs are checked for validity and plausibility. This reduces the probability for entering unreasonable data.

3. Summary of Outputs

Table VII shows a typical table of the complex relative sensitivities for both the theoretical and actual subarray defined for a given MRA. Table VIII shows a table of the theoretical and actual beam pattern data as function of the horizontal angle (in degrees) for both normalized magnitude (from 0 to 1.0) and decibels scale (dB).

Figure C.1 shows a polar plot of the beam pattern in decibels where only the "visible" region is shown, i.e., MRA - 90 degrees to MRA + 90 degrees. The polar plot is the suggested graph type for operational use. Figure C.2 shows a graph of the beam pattern in dB as function of the horizontal angle using rectangular coordinates. Here, the angle varies from - 180 to 180 degrees, since this graph type is used for academic or development purposes. Figure C.3 shows a figure similar to C.2 except that the beam pattern is on a linear scale from 0 to 1.0. The main lobe is much more significant with a linear scale. Figure C.4 shows a polar plot of the beam pattern in decibels for the same case as in Figure C.1 except the mode of operation is "DIFF" instead of "SUM". All graphs possess a legend.

Finally, when a printer (Thinkjet or thermal) is used, an expanded graph can be generated. The expanded graph covers a whole page instead of the normal size which is about half a page. However, the quality of the expanded graph is reduced due to the implementation done by HP. The purpose of this graph is to provide a better curve of the beam pattern. Figure C.5 shows an expanded graph (polar plot) done on a Thinkjet printer.

TABLE VII

COMPLEX RELATIVE SENSITIVITIES OF THE HYDROPHONES

THE REQUESTED MAA = 0 degrees
THE USED MAA = 0 degrees

THE MODE OF OPERATION = SUM

THE COMPLEX RELATIVE SENSITIVITIES USED ARE

HYDRO. NUMBER M , P	DESIGN AMPL X	DESIGN PHASE X	DEGRADED AMPL X	DEGRADED PHASE X
13 -1	1.00000	0.00000	1.00000	0.00000
13 0	1.00000	0.00000	1.00000	0.00000
13 1	1.00000	0.00000	1.00000	0.00000
14 -1	1.00000	0.00000	1.00000	0.00000
14 0	1.00000	0.00000	1.00000	0.00000
14 1	1.00000	0.00000	1.00000	0.00000
15 -1	1.00000	0.00000	1.00000	0.00000
15 0	1.00000	0.00000	1.00000	0.00000
15 1	1.00000	0.00000	1.00000	0.00000
16 -1	1.00000	0.00000	1.00000	0.00000
16 0	1.00000	0.00000	1.00000	0.00000
16 1	1.00000	0.00000	1.00000	0.00000
17 -1	1.00000	0.00000	1.00000	0.00000
17 0	1.00000	0.00000	1.00000	0.00000
17 1	1.00000	0.00000	1.00000	0.00000
18 -1	1.00000	0.00000	1.00000	0.00000
18 0	1.00000	0.00000	1.00000	0.00000
18 1	1.00000	0.00000	1.00000	0.00000
19 -1	1.00000	0.00000	1.00000	0.00000
19 0	1.00000	0.00000	1.00000	0.00000
19 1	1.00000	0.00000	1.00000	0.00000
20 -1	1.00000	0.00000	1.00000	0.00000
20 0	1.00000	0.00000	1.00000	0.00000
20 1	1.00000	0.00000	1.00000	0.00000
21 -1	1.00000	0.00000	1.00000	0.00000
21 0	0.00000	0.00000	0.00000	0.00000
21 1	1.00000	0.00000	1.00000	0.00000
22 -1	1.00000	0.00000	1.00000	0.00000
22 0	1.00000	0.00000	1.00000	0.00000
22 1	1.00000	0.00000	1.00000	0.00000
23 -1	1.00000	0.00000	1.00000	0.00000
23 0	1.00000	0.00000	1.00000	0.00000
23 1	1.00000	0.00000	1.00000	0.00000
24 -1	1.00000	0.00000	1.00000	0.00000
24 0	1.00000	0.00000	1.00000	0.00000
24 1	1.00000	0.00000	1.00000	0.00000
25 -1	0.00000	0.00000	0.00000	0.00000
25 0	1.00000	0.00000	1.00000	0.00000
25 1	0.00000	0.00000	0.00000	0.00000
26 -1	0.00000	0.00000	0.00000	0.00000
26 0	1.00000	0.00000	1.00000	0.00000
26 1	0.00000	0.00000	0.00000	0.00000
27 -1	0.00000	0.00000	0.00000	0.00000
27 0	1.00000	0.00000	1.00000	0.00000
27 1	0.00000	0.00000	0.00000	0.00000
28 -1	0.00000	0.00000	0.00000	0.00000
28 0	1.00000	0.00000	1.00000	0.00000
28 1	0.00000	0.00000	0.00000	0.00000
29 -1	1.00000	0.00000	1.00000	0.00000
29 0	1.00000	0.00000	1.00000	0.00000
29 1	1.00000	0.00000	1.00000	0.00000
30 -1	1.00000	0.00000	1.00000	0.00000
30 0	1.00000	0.00000	1.00000	0.00000
30 1	1.00000	0.00000	1.00000	0.00000
31 -1	1.00000	0.00000	1.00000	0.00000
31 0	1.00000	0.00000	1.00000	0.00000
31 1	1.00000	0.00000	1.00000	0.00000
32 -1	1.00000	0.00000	1.00000	0.00000
32 0	1.00000	0.00000	1.00000	0.00000
32 1	1.00000	0.00000	1.00000	0.00000
33 -1	1.00000	0.00000	1.00000	0.00000
33 0	1.00000	0.00000	1.00000	0.00000
33 1	1.00000	0.00000	1.00000	0.00000
34 -1	1.00000	0.00000	1.00000	0.00000
34 0	1.00000	0.00000	1.00000	0.00000
34 1	1.00000	0.00000	1.00000	0.00000
35 -1	1.00000	0.00000	1.00000	0.00000
35 0	1.00000	0.00000	1.00000	0.00000
35 1	1.00000	0.00000	1.00000	0.00000
36 -1	1.00000	0.00000	1.00000	0.00000
36 0	1.00000	0.00000	1.00000	0.00000
36 1	1.00000	0.00000	1.00000	0.00000
37 -1	1.00000	0.00000	1.00000	0.00000
37 0	1.00000	0.00000	1.00000	0.00000
37 1	1.00000	0.00000	1.00000	0.00000
38 -1	1.00000	0.00000	1.00000	0.00000
38 0	1.00000	0.00000	1.00000	0.00000
38 1	1.00000	0.00000	1.00000	0.00000
39 -1	1.00000	0.00000	1.00000	0.00000
39 0	1.00000	0.00000	1.00000	0.00000
39 1	1.00000	0.00000	1.00000	0.00000
40 -1	1.00000	0.00000	1.00000	0.00000
40 0	1.00000	0.00000	1.00000	0.00000
40 1	1.00000	0.00000	1.00000	0.00000

TABLE VIII

THEORETICAL AND ACTUAL ARRAY BEAM PATTERN DATA

THE REQUESTED MRA = 0 degrees
THE USED MRA = 0 degrees

THE MODE OF OPERATION = SUM

The execution time to compute the beam pattern was 885.490020752 SEC

*** COMPUTED MAGNITUDE BEAM PATTERN DATA ***

ANGLE degrees	THEORETICAL Normalized	dB	ACTUAL Normalized	dB
-180	.15053	-16.45	.15053	-16.45
-170	.03805	-28.39	.03805	-28.39
-160	.03537	-29.03	.03537	-29.03
-150	.05104	-25.84	.05104	-25.84
-140	.01575	-36.06	.01575	-36.06
-130	.08801	-21.11	.08801	-21.11
-120	.09196	-20.73	.09196	-20.73
-110	.05082	-25.88	.05082	-25.88
-100	.03116	-30.13	.03116	-30.13
-90	.08945	-20.97	.08945	-20.97
-80	.12723	-17.91	.12723	-17.91
-70	.12758	-17.88	.12758	-17.88
-60	.21038	-13.54	.21038	-13.54
-50	.34189	-9.32	.34189	-9.32
-40	.42620	-7.41	.42620	-7.41
-30	.42100	-7.51	.42100	-7.51
-20	.31659	-9.99	.31659	-9.99
-10	.31617	-10.00	.31617	-10.00
0	1.00000	0.00	1.00000	0.00
10	.30825	-10.22	.30825	-10.22
20	.31570	-10.01	.31570	-10.01
30	.39467	-8.08	.39467	-8.08
40	.42581	-7.42	.42581	-7.42
50	.33643	-9.46	.33643	-9.46
60	.18888	-14.48	.18888	-14.48
70	.12479	-18.08	.12479	-18.08
80	.12321	-18.19	.12321	-18.19
90	.08713	-21.20	.08713	-21.20
100	.03028	-30.38	.03028	-30.38
110	.04359	-27.21	.04359	-27.21
120	.08020	-21.92	.08020	-21.92
130	.07871	-22.08	.07871	-22.08
140	.01573	-36.07	.01573	-36.07
150	.06156	-24.21	.06156	-24.21
160	.03773	-28.47	.03773	-28.47
170	.06287	-24.03	.06287	-24.03
180	.15053	-16.45	.15053	-16.45

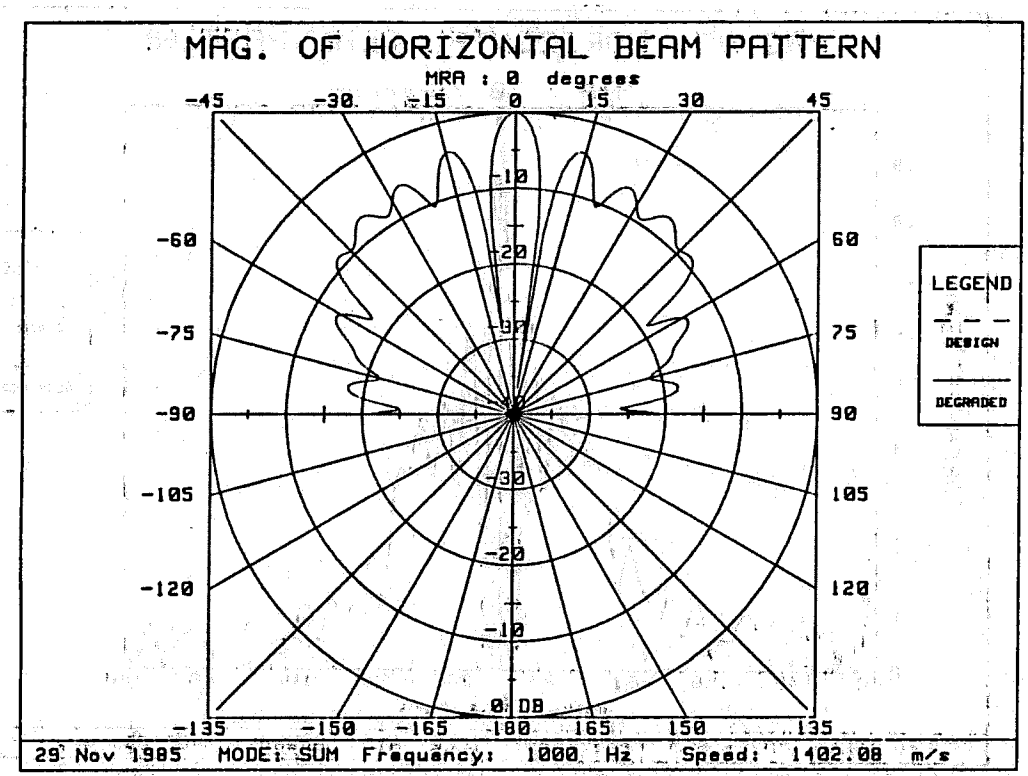

Figure C.1 Polar Plot of Beam Pattern in dB.

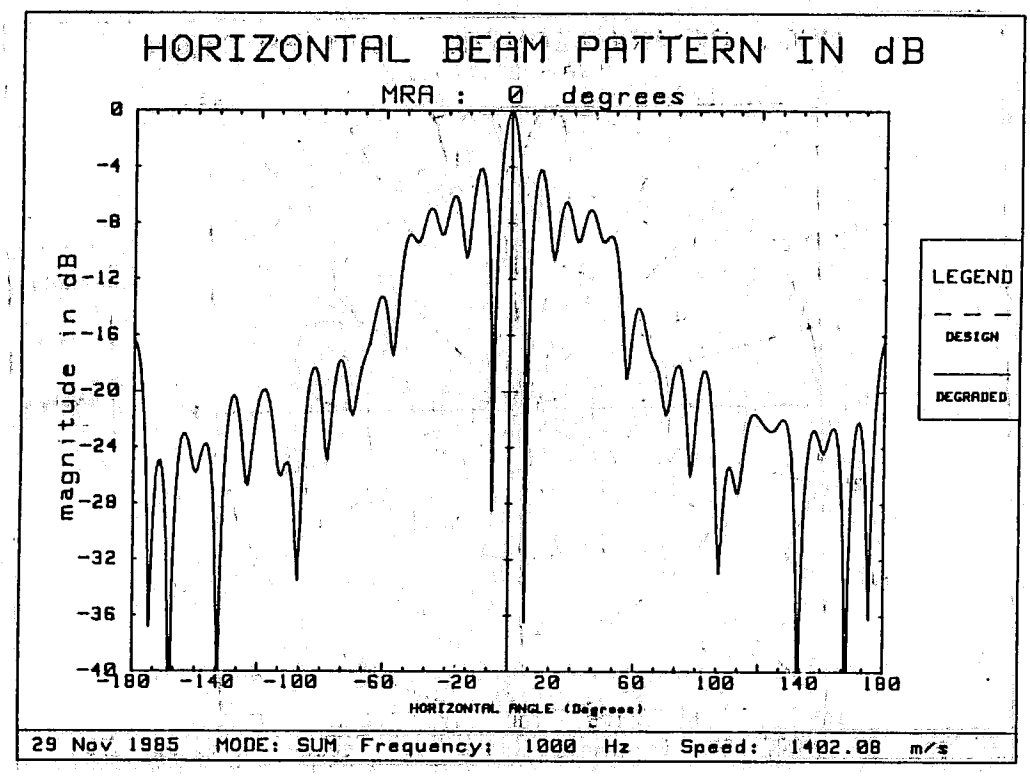

Figure C.2 Magnitude of Beam Pattern in dB.

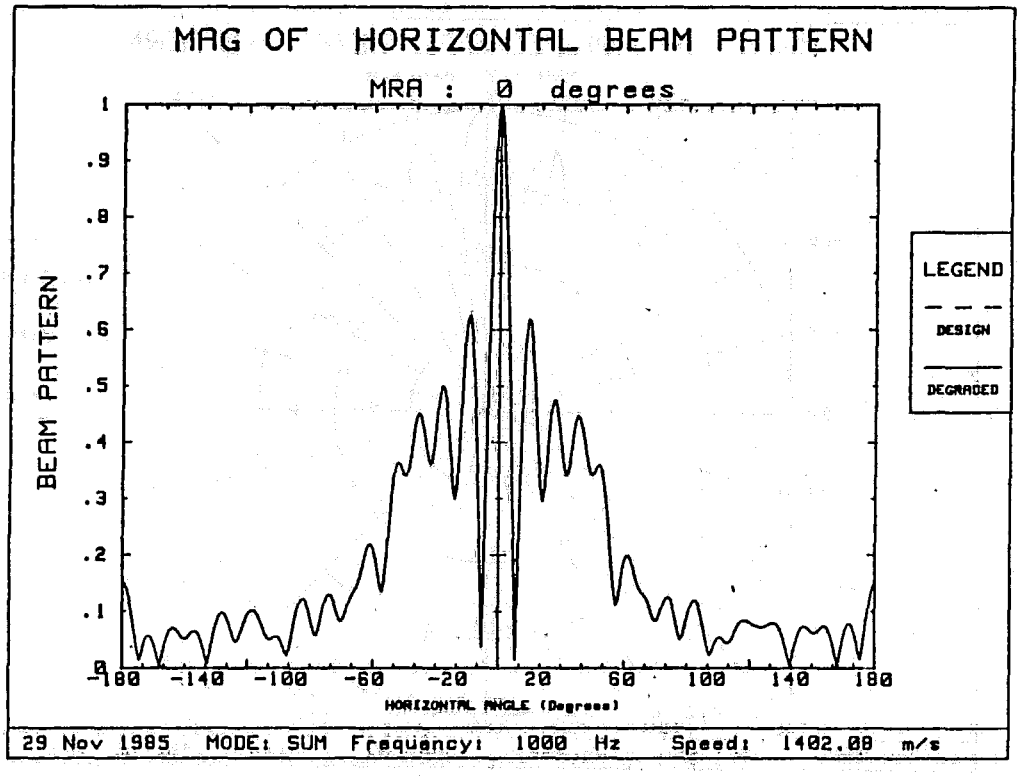

Figure C.3 Normalized Magnitude of Beam Pattern.

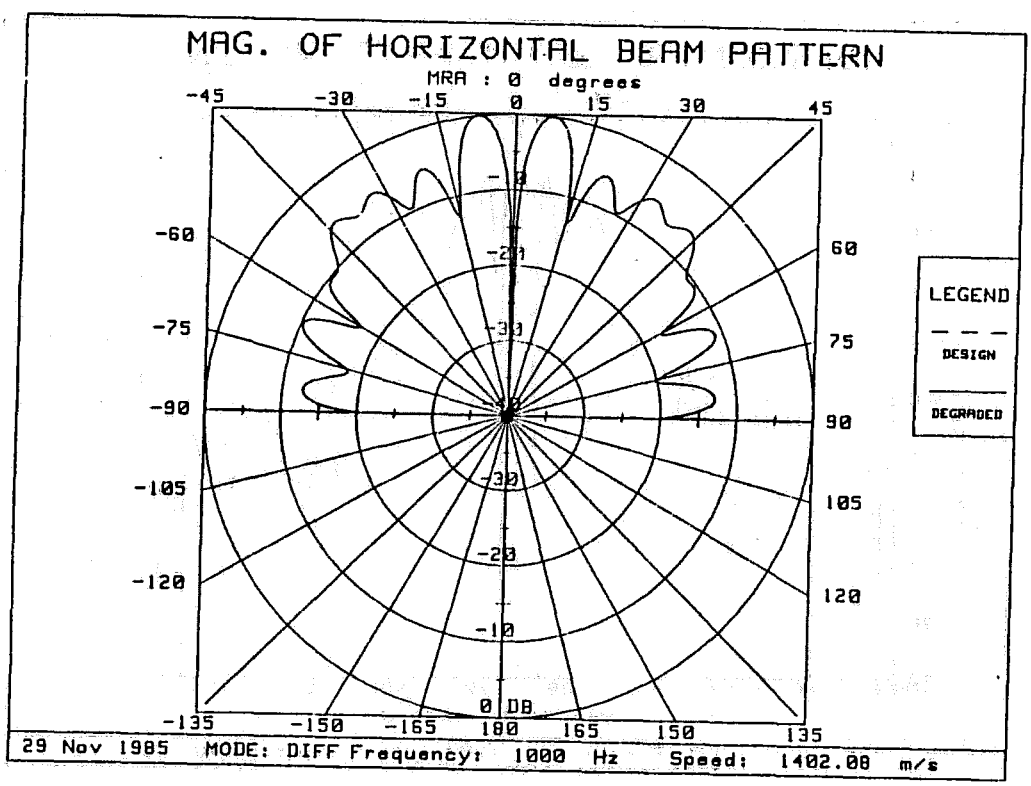

Figure C.4 Polar Plot of Beam Pattern with "DIFF" Mode.

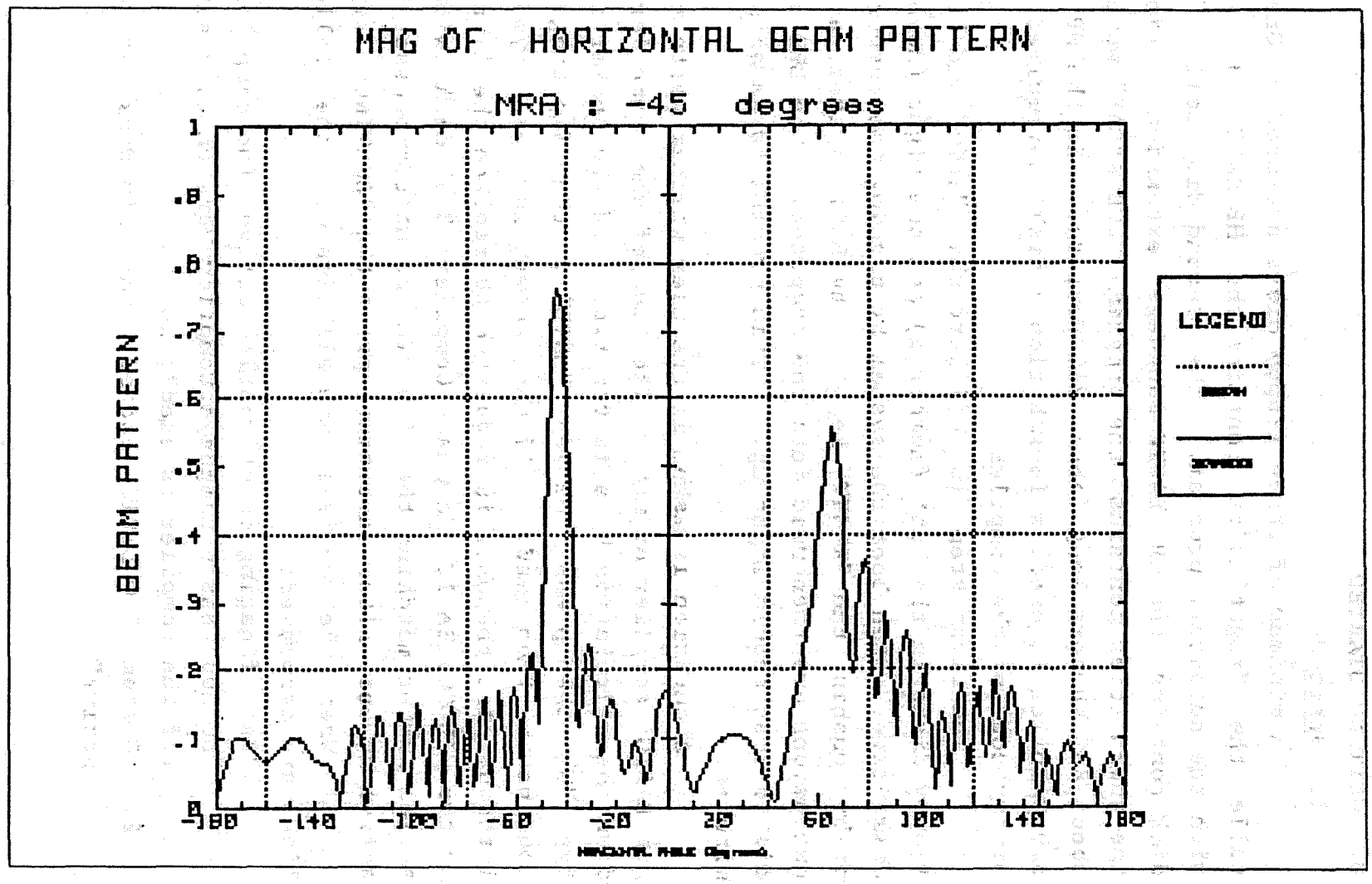

Figure C.5 Expanded Printed Polar Plot of Beam Pattern.

B. GETTING STARTED

1. Start UP

A minimum of three floppy disks are used. One disk called the "right" disk contains the HP BASIC 3.0 system with the autostart program. The second disk called "left" disk contains the HP BASIC language extension. Finally, the third disk contains the program "SUB_ARRAY" which is about 850 sectors long (about 210 Kbytes). If the HP BASIC 3.0 Compiler is used, a fourth floppy disk is required: the binary code of the compiler.

The first step is to insert the "left" and "right" floppy disks in the left (number 1) and the right (number 0) disk drives, then boot the system by turning the computer on or by pushing the "RESET" key. The BASIC system and its extensions are then read into the computer. The LED of the disk drives should light up. This process takes about two minutes.

A question is asked if the user whether wants to use the BASIC compiler or not. The use of the compiler cuts down the calculation by a factor two. If the answer is no, type "N" then press the key "ENTER", the floating point routine is then read. If the answer is yes, type "Y" instead, next the user is requested to replace the left disk by the "BC203 BASIC Language Compiler 3.0" disk with the serial number matching the floating point coprocessor board serial number. After insertion, press the "CONTINUE" key.

When the routines are all loaded, the following lines are displayed:

1. "Fast floating point routine active" (no compiler) or "BC 203 BASIC Language Compiler Version 3.0 active" (if the compiler is used"; and
2. "Custom 3.0 BASIC booted and left disk drive is active".

The main program "SUB_ARRAY" can then be entered in memory. Remove all disks and insert the disk "SUB_ARRAY" in the left disk drive. Next type: Load "SUB_ARRAY". Then press the key "EXECUTE". After a minute or two the program is loaded. This is indicated when the disk driver stops rotating.

One last step, if the compiler is used, type "COMPILE" then press "EXECUTE" to have the program transformed into compiled code.

The program is now ready to be run. Remove all disks, and simply press the key "RUN" to execute the program. The program can be stopped at any time by pressing the key "PAUSE" and then continued by pressing the key "CONTINUE". This does not apply when a peripheral is used, i.e., a printer or a plotter. To stop a peripheral and the program, press the key "STOP". The program can always be re-executed by pressing the key "RUN"

2. Input, Menus, and Options

The first message shown to the user is Figure C.6 which provides the title and purpose of the program. Press "CONTINUE" when done. If the date and the time need to be entered, the user will be requested to do so by the message in Figure C.7 This is done only once when the program is run for the first time after booting.

Statements of what is being executed are shown at the bottom of the screen, such as, hydrophone locations being read, when the program is running, etc. A question about wether real data values of actual relative sensitivities from disk file are to be used instead of the default values, is then asked. If the answer is "Y", insert the disk "FREQ" in the right disk driver. Then the first menu is provided as in Figure C.8 to change the basic parameters of the program and to choose the outputs. Softkeys are used to change the default values. The following list the function of the keys:

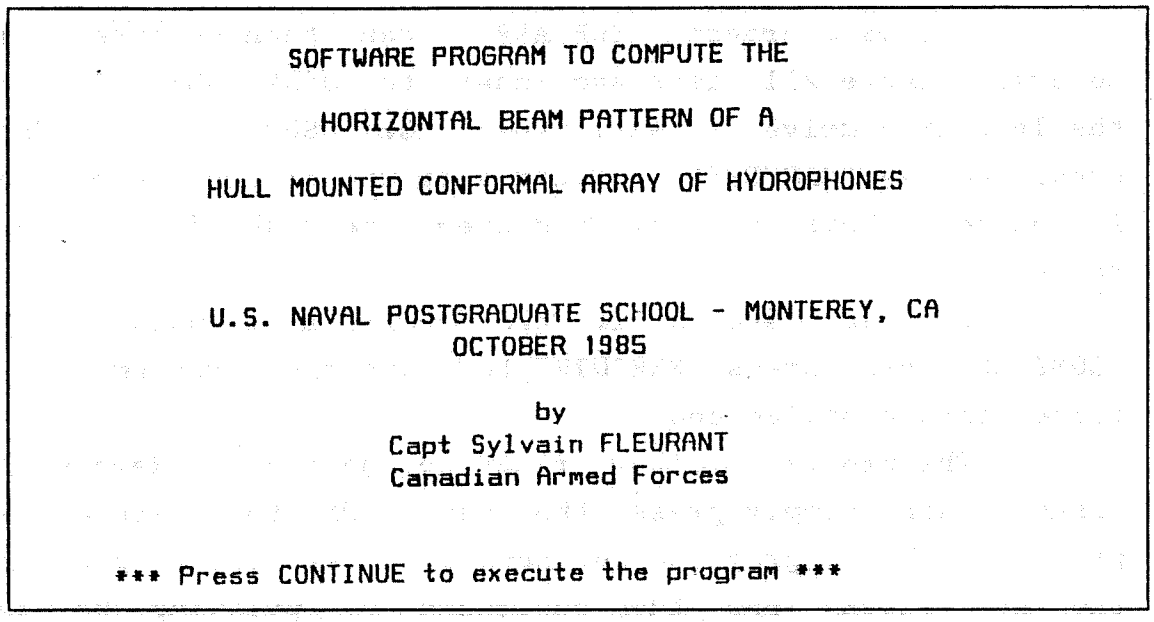

Figure C.6 Program Title.

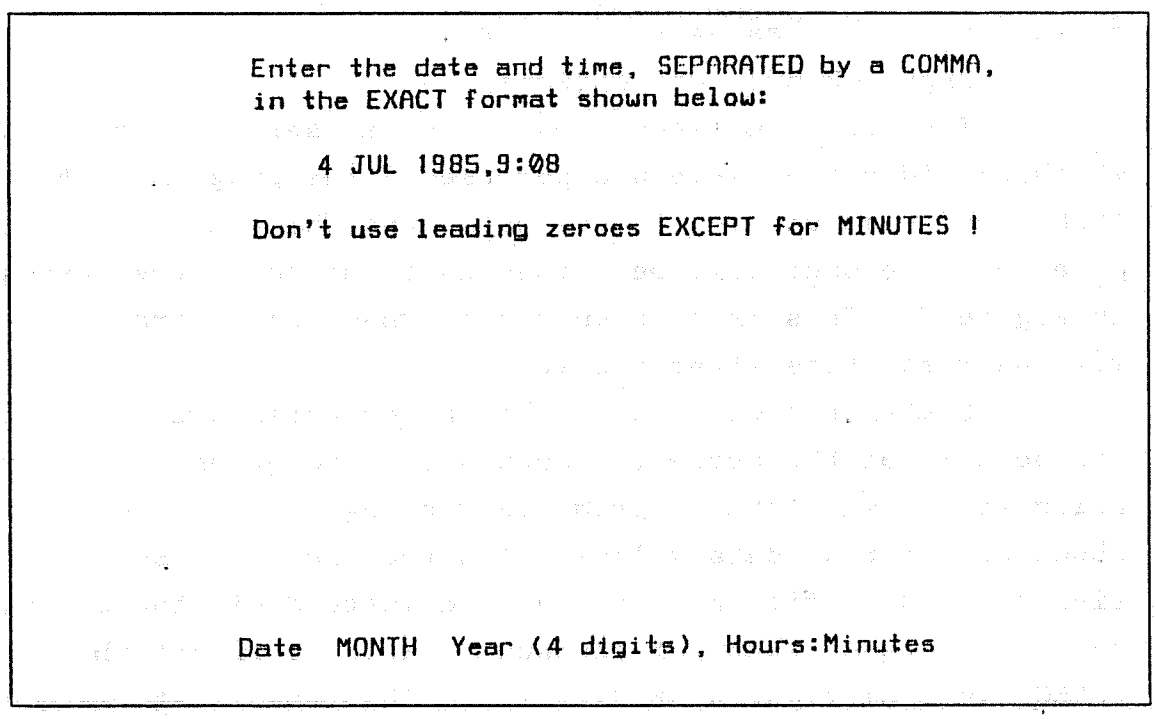

Figure C.7 Change Date and Time.

1. Key 0: Output a table of the theoretical and actual

array complex relative sensitivities of the hydrophones used for a given MRA;

2. Key 1: Output a table of the beam pattern data as function of the horizontal angle;
3. Key 2, 3, and 4: For the three graph types;
4. Key 5: Frequency in Hz;
5. Key 6: MRA in degrees between - 180 to 180 degrees;
6. Key 7: Speed of sound in m/sec from 1400 to 1600 m/sec;
7. Key 8: Mode - "SUM" or "DIFF"; and
8. Key 9: Start execution of the rest of the program.

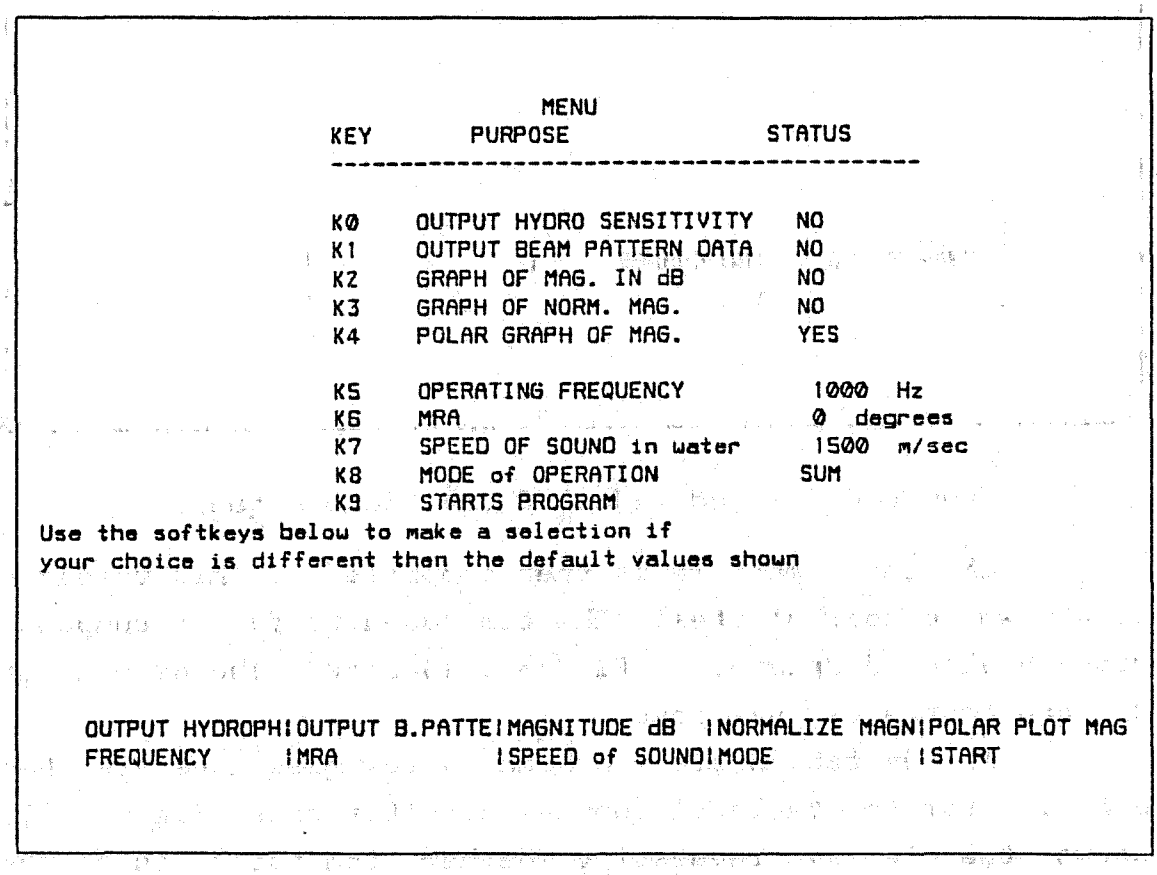

Figure C.8 Main Menu.

Again, more statements are provided to indicate what is being executed. Afterwards, another menu is shown, see

Figure C.9, asking if the user wants to change the status of any staves or hydrophones. Softkeys K0 and K1 are used for input. The next section shows the details of this menu.

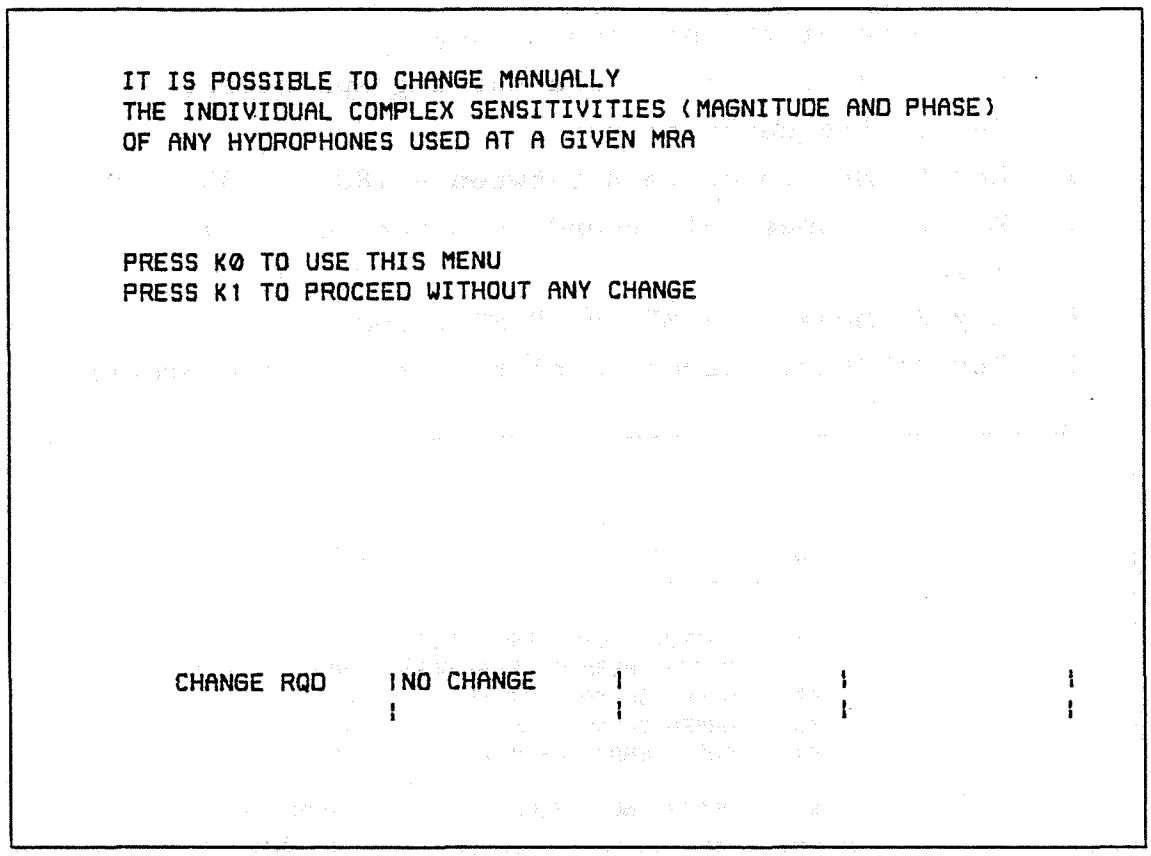

Figure C.9 Individual Change Menu Request.

The beam pattern is then computed. This requires about two to four minutes. The time used to do the computation is also displayed. Figure C.10 shows the display at the end of the calculations.

If any tables are required, a few questions are then asked. For the table of complex sensitivities, Figure C.11 shows the display requesting whether the table is to be displayed or printed. Similarly, for the table of beam patten data, Figure C.12 indicates the request for the steps of the horizontal angles to be used, e.g., every five or ten

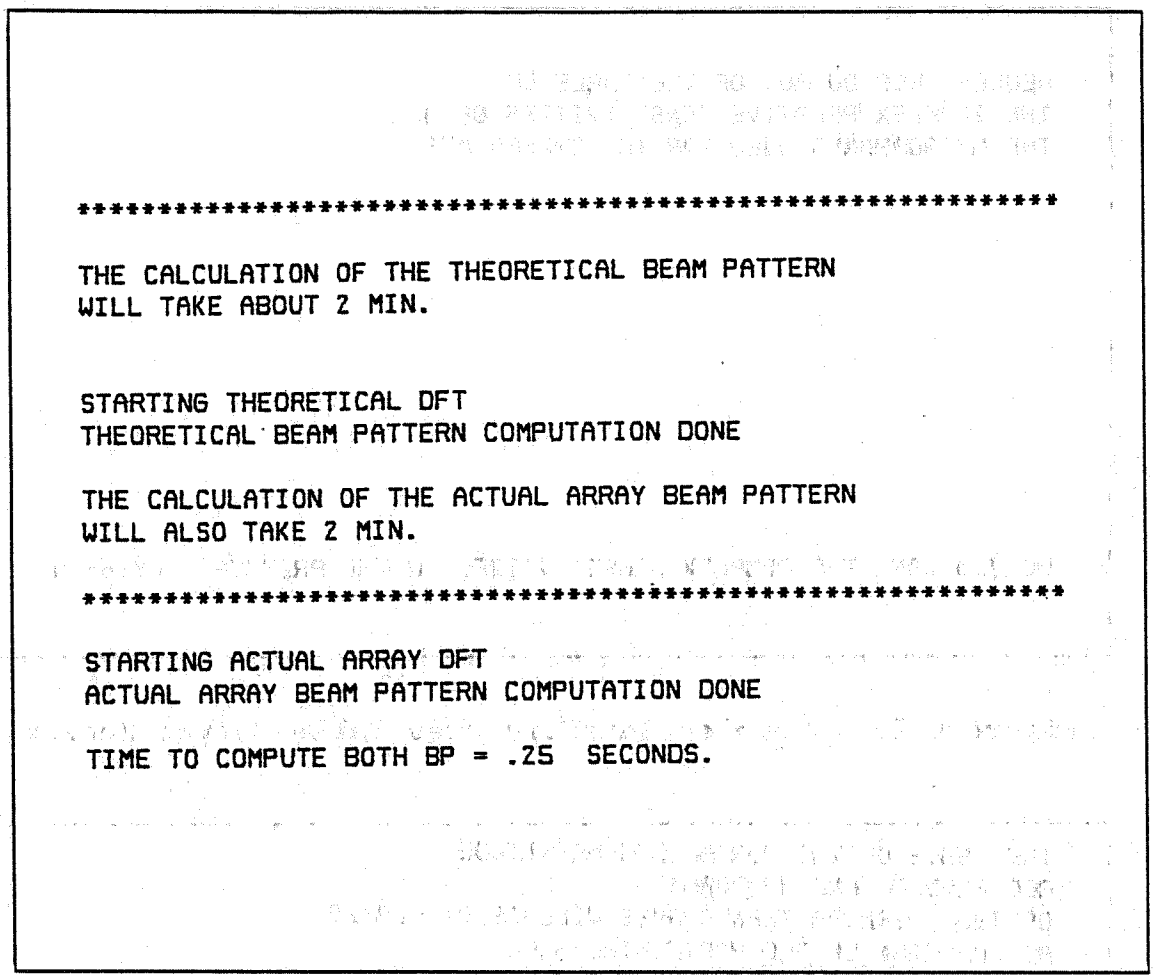

Figure C.10 Beam Pattern Calculations.

degrees. A choice of the medium, CRT or printer is also requested. Then the table is cleared. It is possible to pause the program (press "PAUSE") to scroll the data with the knob if the table was displayed on the CRT.

If no graphs were requested, a message indicating this fact is shown for about five seconds, then the program stops.

If any graphs were requested, the following procedure is done for every graph. The graph is first displayed for two seconds, next a menu is shown requesting the size and medium to provide the graph. Figure C.13 shows the menu

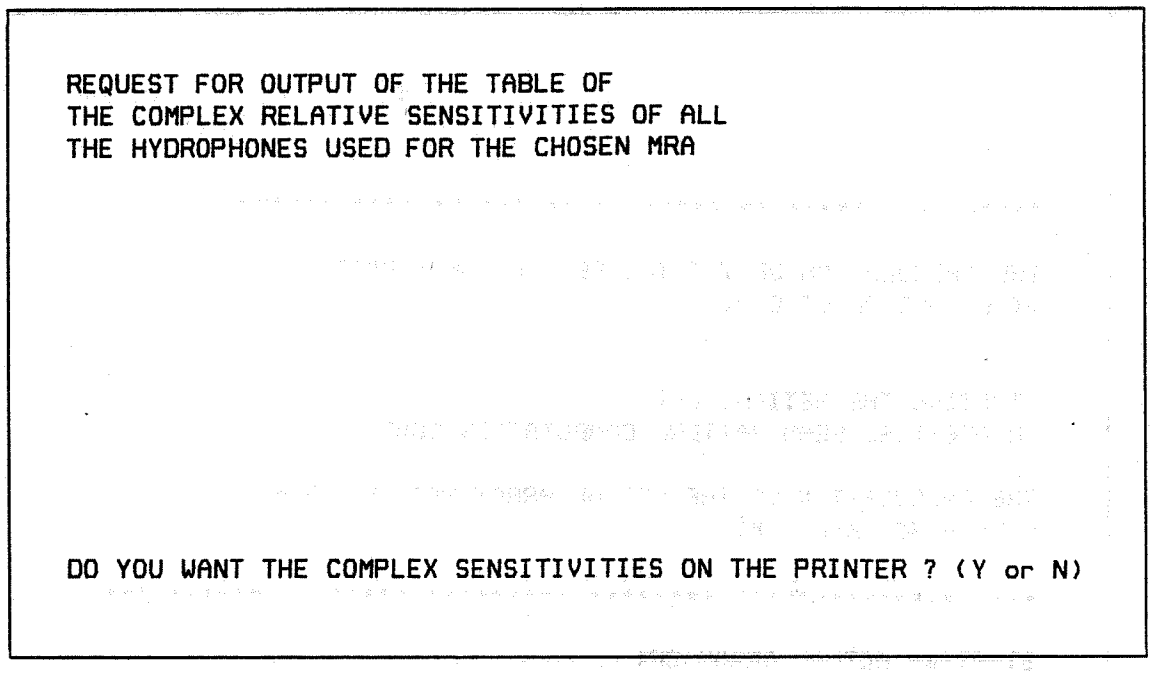

Figure C.11 Complex Sensitivities Table Output Medium.

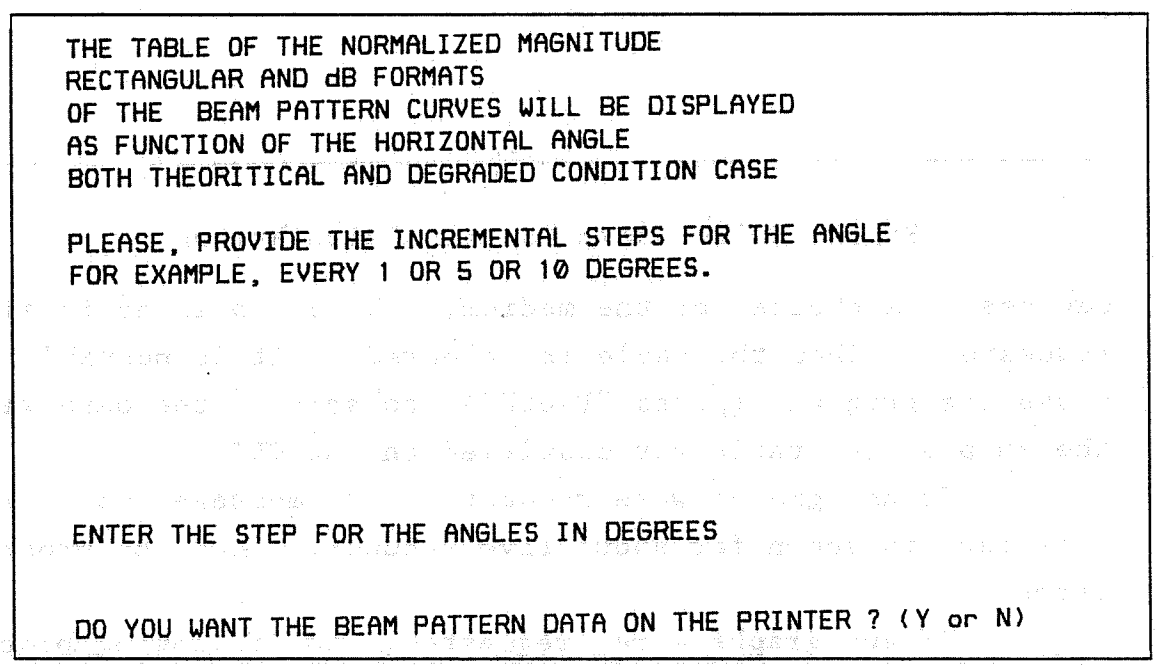

Figure C.12 Beam Pattern Data Output.

with the options. Note: Toggle (key 0) allows the user to

look at the graph and then to return to the menu to make a choice. Both the knob on the keyboard and key 3 and key 4 can be used to move the cursor (the arrow =>) to a selection.

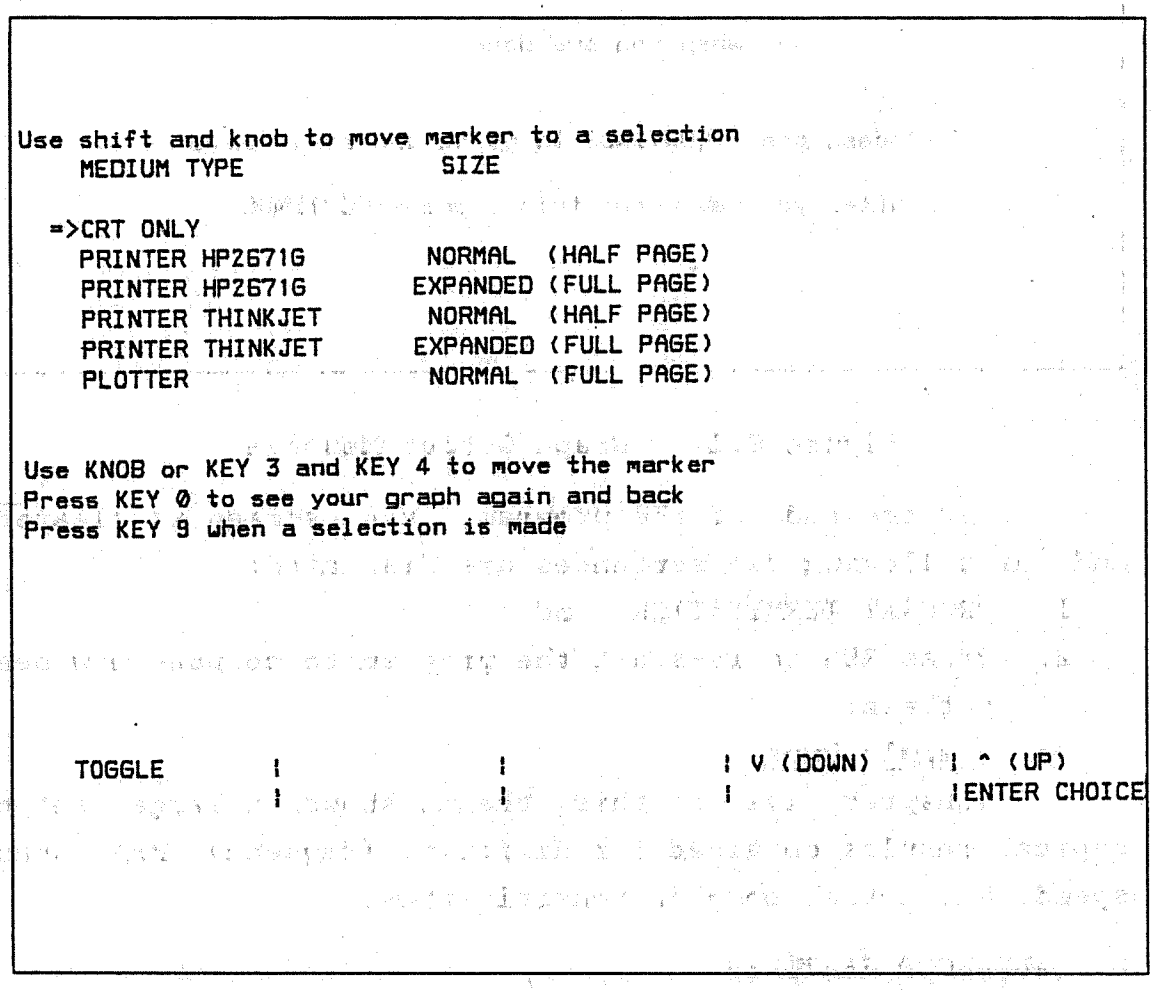

Figure C.13 Output Medium and Size Menu.

After a selection is made, Figure C.14 is displayed. It indicates that after the graph has been output, the user can go to the next features (next graph or end of the program) by simply pressing the key "CONTINUE".

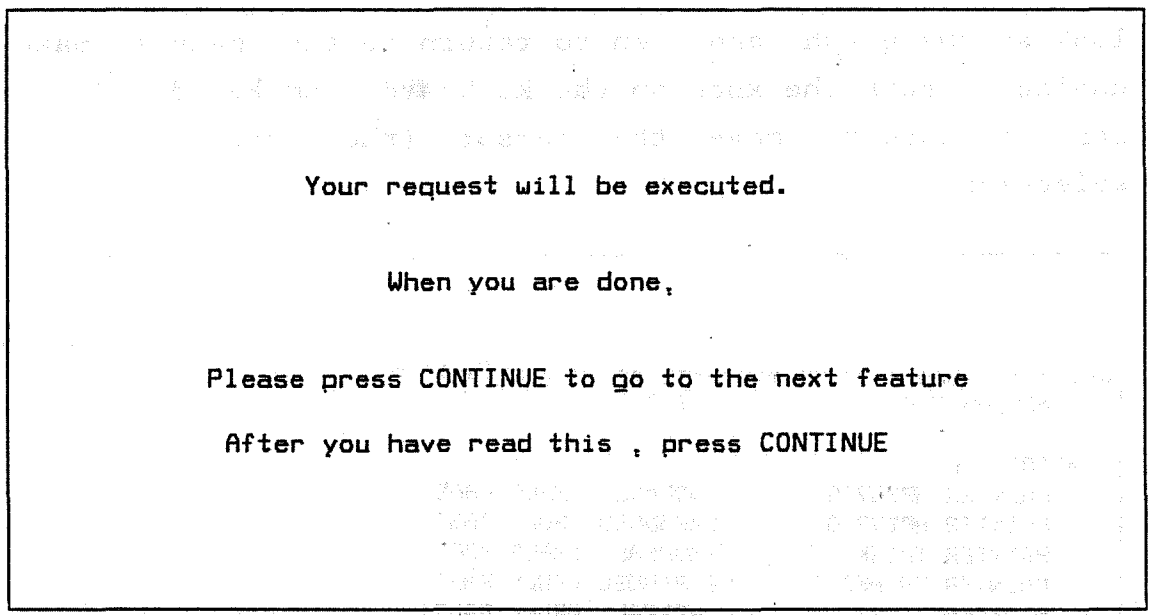

Figure C.14 Graph Output Message.

At the end of the program, the screen is cleared, and the following two sentences are displayed:

1. PROGRAM TERMINATION; and
2. Press RUN to re-start the program to compute new beam pattern.

3. Sample Runs

Chapter five of this thesis shows a large set of typical results obtained for different frequency, MRA, sound speed, and actual complex sensitivities.

C. ADVANCED FEATURES

1. Menu for Individual Stave Changes

The second menu allows the user to change the status of any hydrophones used at a given MRA. The following choice is available:

1. Disconnect a stave or any of its hydrophones;
2. Pad a stave with up to three capacitors;
3. Specify the magnitude and phase of the sensitivity of any hydrophone; and

4. Exit when done.

All the choices are also provided with a sub-menu where the option "exit" (key 9) can be used to get out at any time, even if no changes are required.

Figure C.15 shows first display if the second menu is used. It shows the staves that can be changed and request which one to change. Enter a choice. Next a sub-menu is provided (Figure C.16) to request one of the options of the previous paragraph.

If "disconnect" is chosen, the sub-menu of Figure C.17 is provided. If padding is chosen, the sub-menu of Figure C.18 is provided. Moreover, if "individual complex weights change" is chosen, the sub-menu of Figure C.19 is displayed, where a hydrophone is to be chosen. Then another sub-menu (Figure C.20) requests the new amplitude and/or phase sensitivity of the hydrophone. The previous and new values of the complex sensitivity are shown. More an hydrophones of the selected stave can be chosen. The user may return to the main menu by exiting with the key 9 (EXIT).

Again, the selection of staves is provided to the user and a stave number is requested. Proceed if you want more changes, or simply enter any valid value and exit (with the following sub-menu) to proceed to the calculation of the beam pattern.

2. Linear Graph and Expanded Graph

As mentioned earlier, when a graph is printed on a thermal or a Thinkjet printer, two options of size are available: normal and expanded. The normal size covers only half a page, while the expanded one covers a full page. However, the clarity of the printing is poor as showhn in Figure C.5 Nevertheless, the expanded graph allows a better reading of the curves.

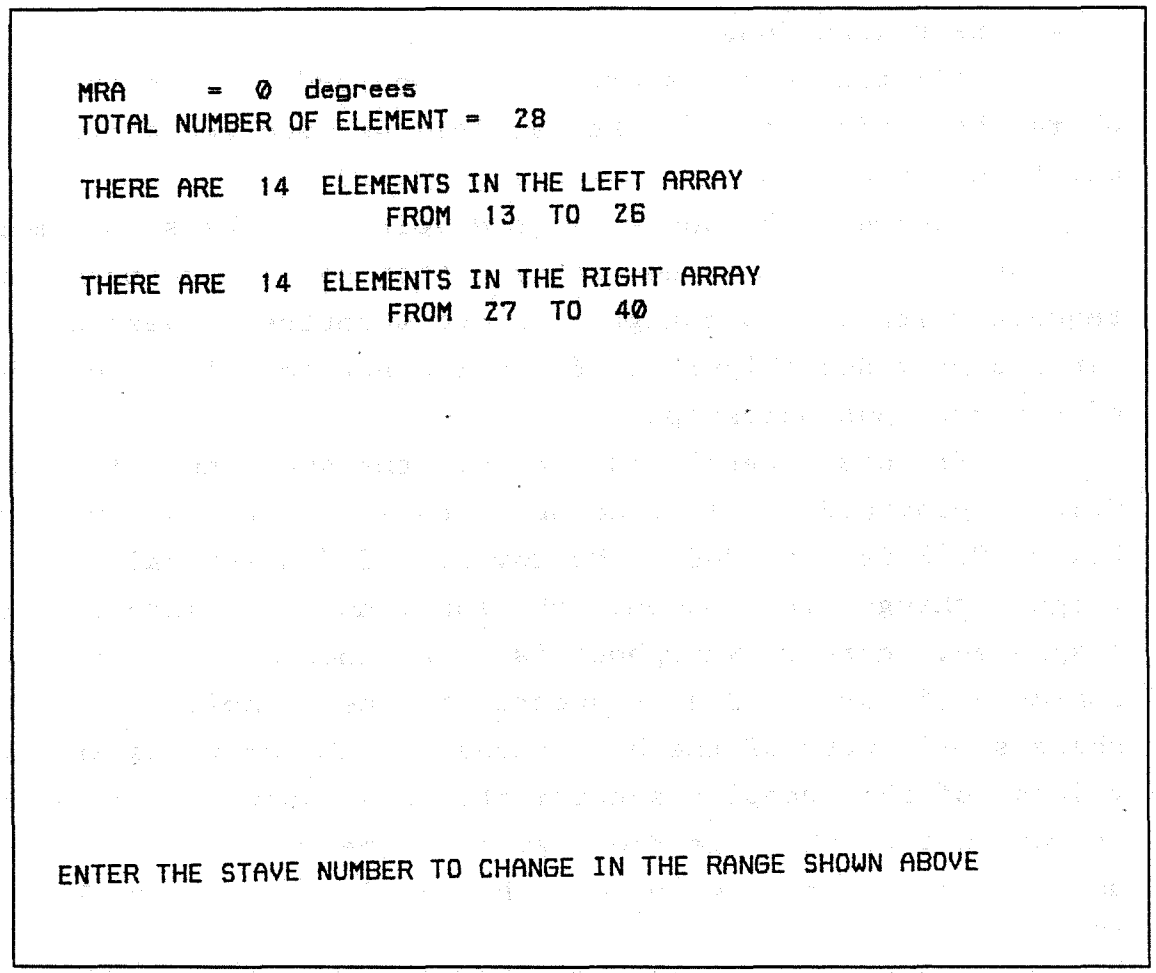

Figure C.15 Stave Choice for Changes.

D. PROGRAM LIMITATIONS.

The program was implemented for one class of submarine, with a limited set of real time delays defined for only one sound speed. MRA's are available in increments of five degrees. If the operator choose an intermediate MRA, the program will round off to the nearest five degrees.

To obtain the second mode of operation, the whole program has to be re-run. The same applies for a new frequency, a new MRA, or a new sound speed. Changes in the second menu are only temporary.

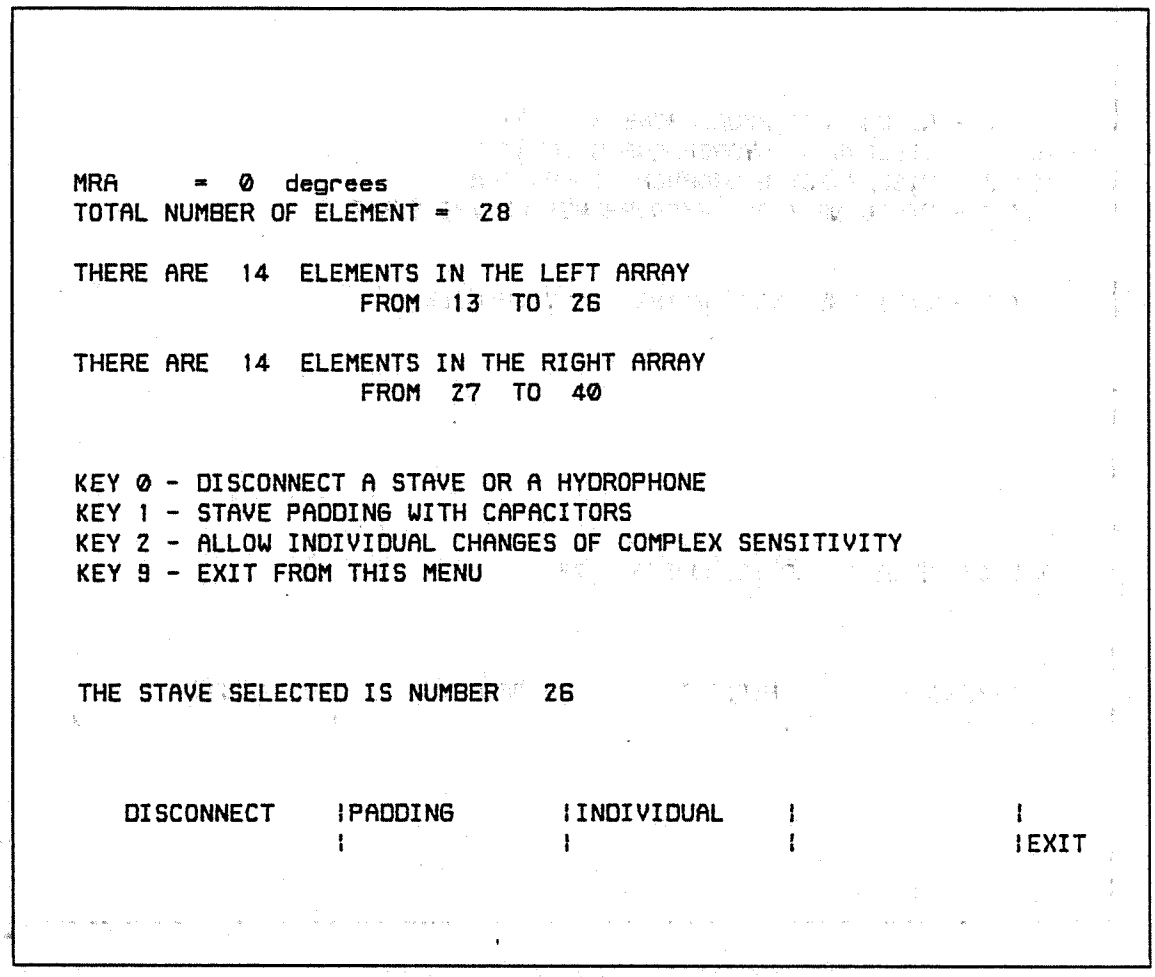

Figure C.16 Options for Individual Status Changes.

The program does not take in consideration the movement of the submarine or the target.

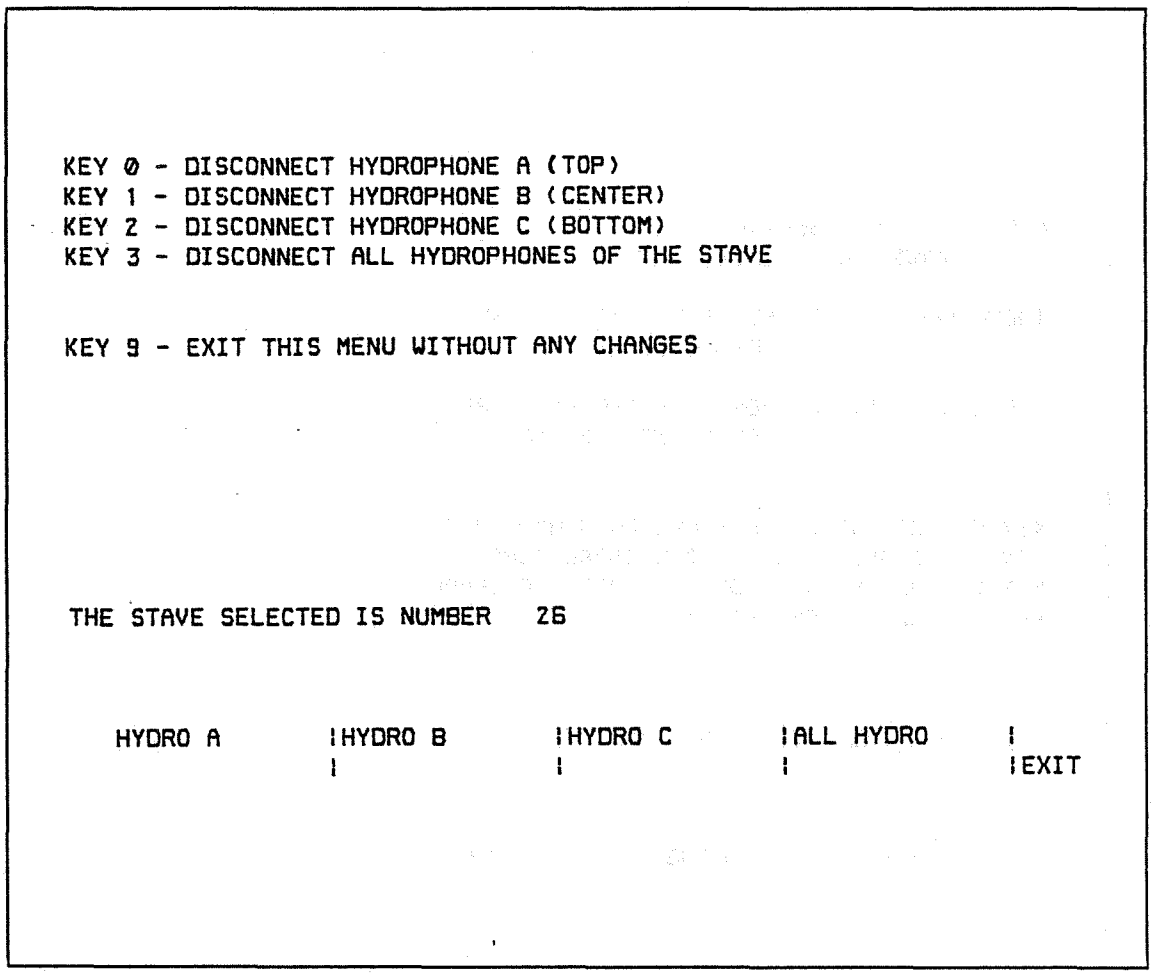

Figure C.17 Disconnect Stave Menu.

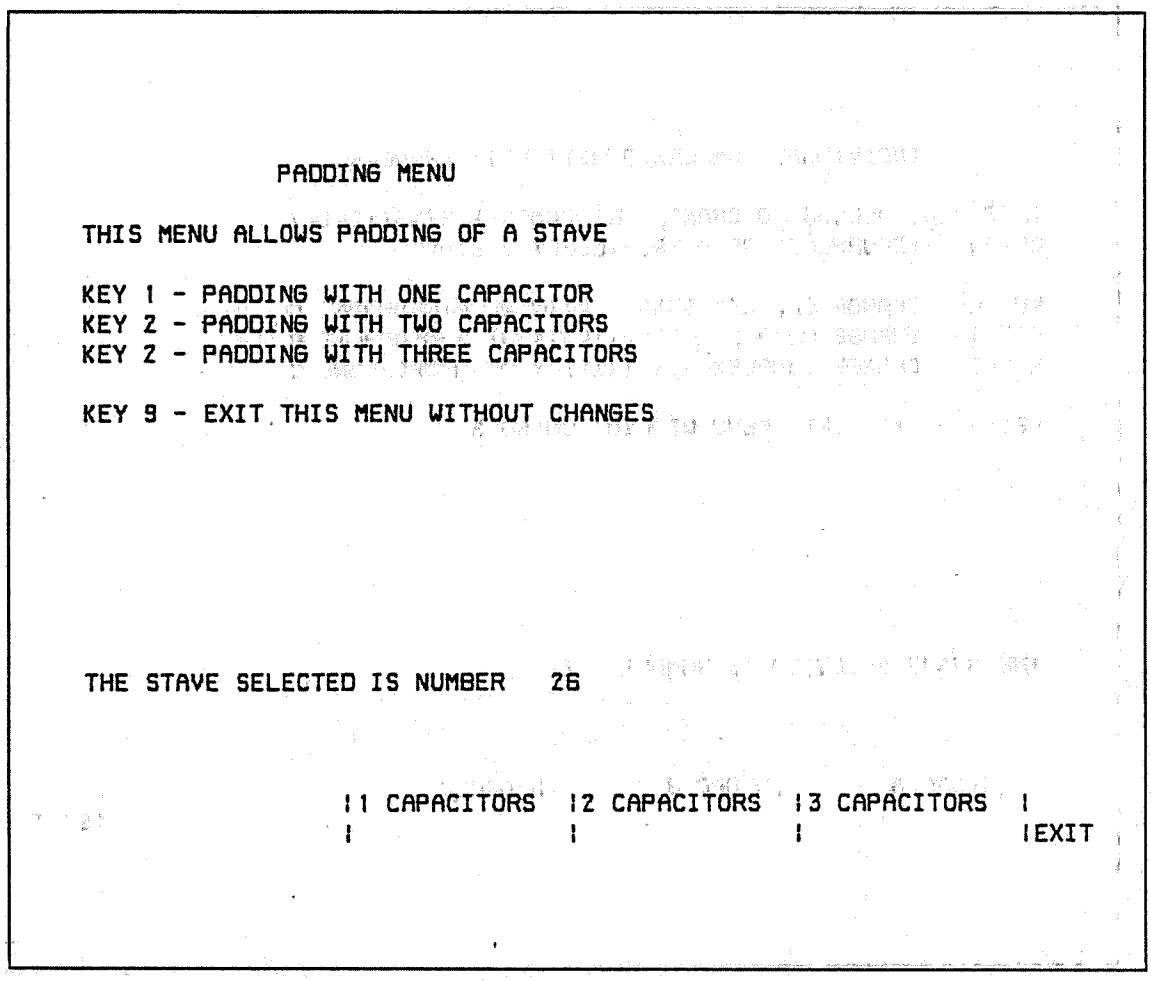

Figure C.18 Pad the Stave Menu.

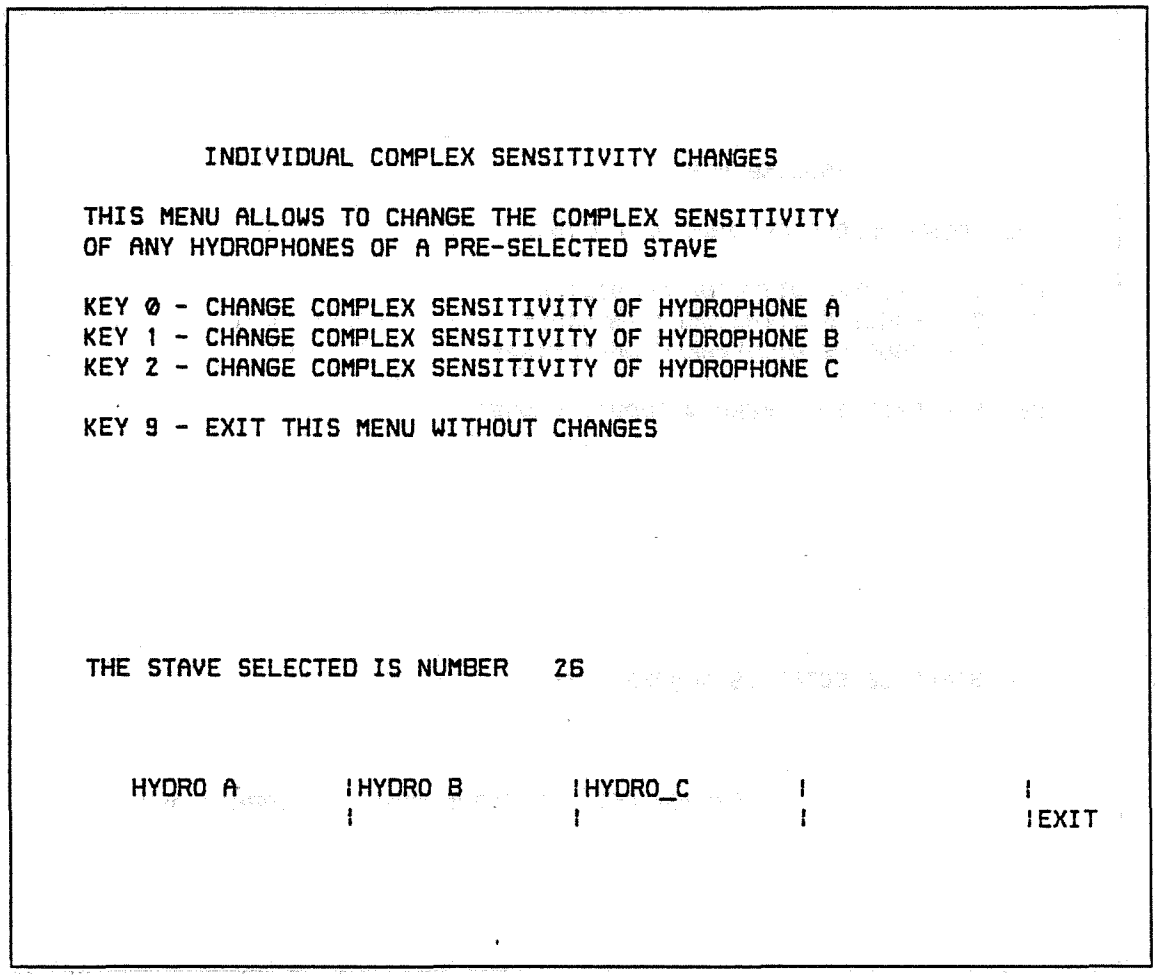

Figure C.19 Complex Sensitivity Changes Menu.

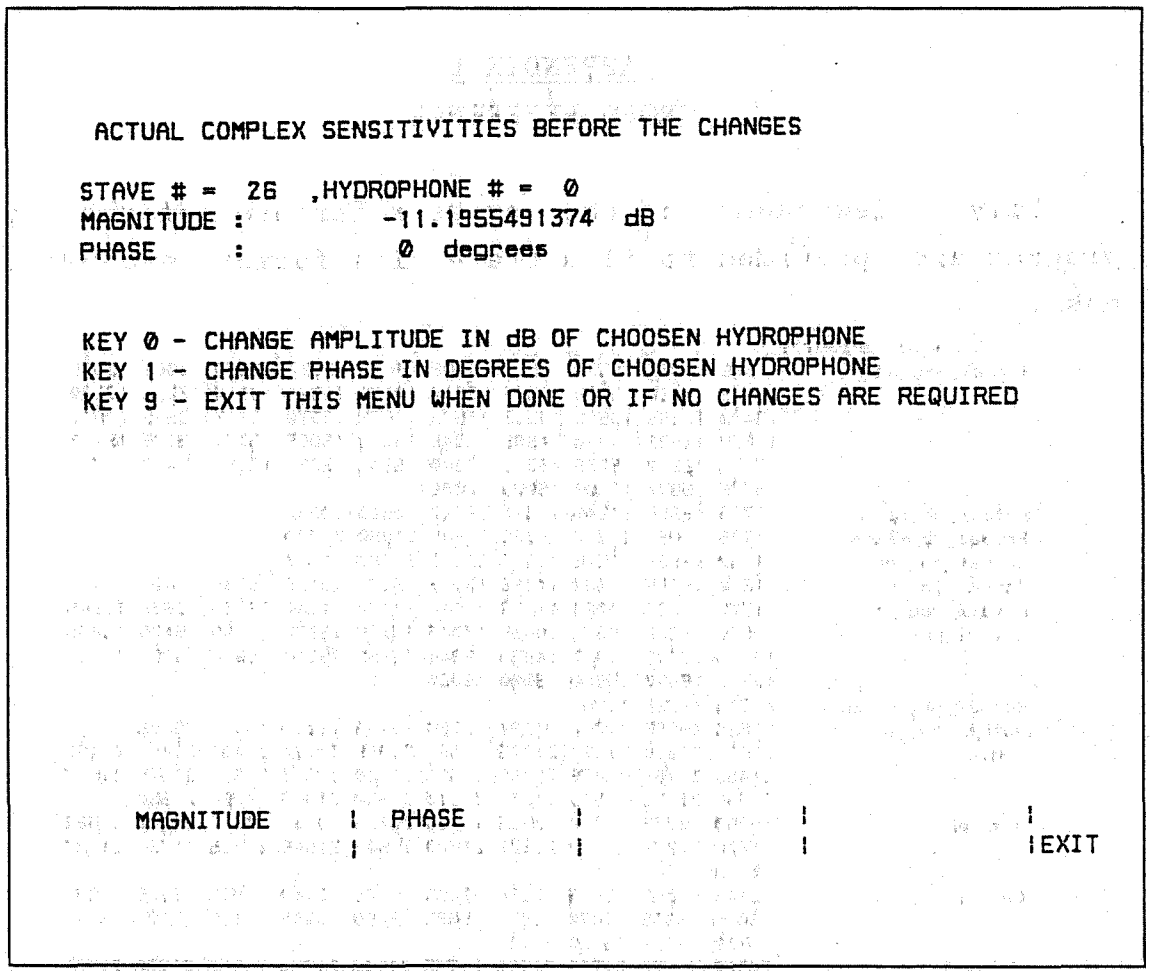

Figure C.20 Change Amplitude and Phase of Hydrophone.

APPENDIX D

CROSS REFERENCE

Only a few pages of the cross reference of the main program are provided to illustrate its format and usefulness.

```
*   Numeric Variables
Actual_amp_wg          1565 <-DEF  8580  8595  9005  9010  9020  9060  9065  9075
                       9115  9120  9130  9835  9850 10385 13040 13110 13225 13280
                      13400 13455 13520 13535 13545 13560 13570 13585 13595 13610
                      13620 14975 14980 14985 15045 15050 15055 15160 15165 15170
                      15215 15220 15225 15335 15340 15345 15390 15395 15400 15725
                      15775 15870 15900 16000 16030
Actual_db_bp           1715 <-DEF 11240 11310 16620 20520 20530
Actual_db_xaxis        1725 <-DEF 11310 21935 21945 21960 21970
Actual_db_yaxis        1730 <-DEF 11310 21935 21945 21960 21970
Actual_im_bp           1690 <-DEF 11060 14985 15055 15170 15225 15345 15400
Actual_mag_bp          1700 <-DEF 11060 11135 11140 11200 11240 16620 19635 19645
Actual_pha_wg          1580 <-DEF  9830 10605 13045 13115 13230 13285 13405 13460
                      13795 13875 13910 14970 15040 15155 15210 15330 15385 15725
                      15775 15870 15900 16000 16030
Actual_phase_bp        1705 <-DEF 11060
Actual_re_bp           1685 <-DEF 11060 14980 15050 15165 15220 15340 15395
Angle                 21315 21330 21370 21375 21380 21395 21410 21430 21440 21450
                      21455 21465 21475 21490 21505 21520 21530 21535 21545 21555
                      21560 21570 21595 21600 21610 21620 21635 21650 21690
Bottom                19360 19375 19440 19515 19790 20265 20280 20345 20400 20680
                      21220 21230 21270 21320 21600 21605 21645 21725 21750 21785
                      22120
Center                 2445 <-DEF  2875  2885  2895  2905  2910  2920  2925  2930
                       3085  5065  5070  5075  5085  5090  5095  5100  5105  5115
                       5120  5125  5130  5135
Oblevel               20900 <-DEF 21725 21730 21735 21740 21750 21755 21760 21765
Freq                   2645 <-DEF  3130  3140  5115  5625  5655  5660  5690  6030
                      14210 14280 14395 14450 14570 14625 14960 15030 15145 15200
                      15320 15375 18730
From_angle_1           2560 <-DEF  3215 21830 21865 21870 21935 21940
From_angle_1_r         2595 <-DEF 21825 21830
From_angle_2           2570 <-DEF  3225 21840 21885 21890 21895 21955 21960 21965
From_angle_2_r         2610 <-DEF 21825 21840
Graph_scale            2635 <-DEF  3205 17060 17255 17450 18420 18470 18505 18675
                      18870 18890 19175 19215 19230 19355 19425 19555 20070 20105
                      20125 20260 20330 20440 20990 21045 21075 21190
High                   6270 <-DEF  6290  6305  6315  6345  6350
Hydro_nn               2490 <-DEF  8240  8455  8485  8515  8545  8575  8580  9535
                       9565  9595  9825  9830  9835  9850 10385 10605
I                      2355 <-DEF  3725  3730  3735  3740  3860  3865  3870  3875
                       4345  4350  4355  4360  4365  4370  4375  4380  4385  4390
                       4395  4400  4430  4435  4440  4445  4450  4455  4460  4465
                       4470  4475  4485  4490  4505  4510  4515  4520  4525  4530
                       4535  4545  4550  4565 10850 10855 10860 11115 11120 11125
                      11135 11140 11150 12100 12125 12130 12205 12210 12230 12235
                      12290 12295 12365 12370 12390 12395 12420 14180 14210 14230
                      14235 14280 14300 14305 14395 14415 14420 14450 14470 14475
                      14570 14590 14595 14625 14645 14650 14695 14930 14960 14980
                      14985 15030 15050 15055 15145 15165 15170 15200 15220 15225
                      15320 15340 15345 15375 15395 15400 15445 16615 16620 16625
J                      2355 <-DEF 11920 11930 11985 11990 11995 12010 12015 12020
                      19595 19600 19605 20480 20485 20490 21265 21270 21275 21870
                      21875 21880 21895 21900 21905
```

```
     >>>>   Subprogram   <<<<
 Z3880    FNIs_string_num$

*   Numeric Variables
Temporary_real        Z4015 <-DEF Z4070 Z4080 Z4110 Z4130

*   String Variables
Data_line$            Z3880 <-DEF Z4035 Z4055 Z4065 Z4070 Z4080 Z4130
Ok_input$             Z4020 Z4030 Z4040 Z4120 Z4135 Z4150

*   String Functions
FNIs_string_num$

Unused entries =    47

     >>>>   Subprogram   <<<<
 Z4180    FNDefine_used_mra

*   Numeric Variables
Mra_value             Z4260 <-DEF Z4280 Z4290 Z4310 Z4325 Z4340 Z4350 Z4375
Requested_mra         Z4180 <-DEF Z4270 Z4275 Z4305 Z4310 Z4320 Z4325 Z4335 Z4340
                      Z4350

*   Numeric Functions
FNDefine_used_mra

Unused entries =     2

     >>>>   Subprogram   <<<<
 Z4410    Comp_axis_pplot

*   Numeric Variables
I                     Z4500 Z4505 Z4510 Z4515 Z4525 Z4530 Z4535 Z4545 Z4550 Z4555
                      Z4565 Z4570 Z4575 Z4585 Z4590 Z4615
Mag_db_bp             Z4410 <-DEF Z4505 Z4530 Z4535 Z4550 Z4555 Z4570 Z4575 Z4585
                      Z4590
Minimum_nn_db         Z4410 <-DEF Z4505 Z4530 Z4535 Z4550 Z4555 Z4570 Z4575 Z4585
                      Z4590
Psi                   Z4410 <-DEF Z4525 Z4530 Z4535 Z4545 Z4550 Z4555 Z4565 Z4570
                      Z4575 Z4585 Z4590
Xaxis_db_bp           Z4410 <-DEF Z4510 Z4530 Z4550 Z4570 Z4585
Yaxis_db_bp           Z4410 <-DEF Z4515 Z4535 Z4555 Z4575 Z4590

*   SUB Subprograms
Comp_axis_pplot

Unused entries =    10

     >>>>   Subprogram   <<<<
 Z4645    Comp_mag_in_db

*   Numeric Variables
I                     Z4745 Z4750 Z4755 Z4765 Z4775
Mag_db_bp             Z4645 <-DEF Z4755 Z4765
Magnitude_bp          Z4645 <-DEF Z4750 Z4765
Minimum_nn_db         Z4645 <-DEF Z4740 Z4755
Minimum_nn_mag        Z4740 Z4750

*   SUB Subprograms
```

APPENDIX E

LISTINGS

```
1000  ! ************************************************************************
1005  ! PROGRAM SUB_ARRAY                    Version 1.0    26 NOVEMBER  1985    *
1010  !                                                                          *
1015  ! PROGRAMMER: SYLVAIN FLEURANT, Capt          DATE: 26 NOVEMBER  1985    *
1020  !             Canadian Armed Forces                                        *
1025  !                                                                          *
1030  ! PURPOSE: HP BASIC program to compute the beam pattern produced by a     *
1035  !          conformal array of hydrophones mounted on the hull of a         *
1040  !          submarine as function of frequency, speed of sound in water,    *
1045  !          configuration, mode, and actual complex sensitivities.          *
1050  !          This is in order to evaluate beam pattern degradation as        *
1055  !          function of individual elements degradation.                    *
1060  !                                                                          *
1065  ! Implementation Language: HP BASIC 3.0                                    *
1070  !                                                                          *
1075  ! Implementation Machine : HP 200's Series Microcomputer                   *
1080  !                          with Floating Point Co-processor.               *
1085  !                                                                          *
1090  !                                                                          *
1095  ! INPUT  : - MRA (Maximum Response Angle (degrees)), frequency (Hz),       *
1100  !            speed of sound in water (m/sec), Mode of operation (SUM,      *
1105  !            DIFF, or both), and files of complex sensitivities.           *
1110  !                                                                          *
1115  ! OUTPUT : - Graphs and tables of theoretical and degraded beam pattern.   *
1120  !                                                                          *
1125  ! DESCRIPTION:                                                             *
1130  !                                                                          *
1135  ! 1.     This program generates beam pattern (or directivity function)    *
1140  ! produced by a conformal array of hydrophones mounted on the hull of a   *
1145  ! specific submarine type.  The beam patterns can be expressed in three    *
1150  ! (3) formats:                                                             *
1155  !                                                                          *
1160  !           i) Plot of the normalized magnitude of the beam pattern as     *
1165  !              function of the horizontal angle (psi) in degrees from      *
1170  !              -180' to 180' where 0' correspond to the nose orientation   *
1175  !              of the submarine,                                           *
1180  !          ii) A similar plot in decibel, and                              *
1185  !         iii) Polar plot in dB using horizontal angle.                    *
1190  !                                                                          *
1195  ! Both theoretical (not degraded) and degraded beam patterns are          *
1200  ! available.                                                              *
1205  !                                                                          *
1210  ! 2.     A choice of frequency (in Hertz), of speed of sound in meters     *
1215  ! per seconds, of actual MRA (Maximum Response Angle (in degrees)), and   *
1220  ! Mode of operation (SUM, DIFF, or both) have to be selected in order to  *
1225  ! generate a specific beam pattern.                                       *
1230  !                                                                          *
1235  ! 3.     A complete User-Friendly Menu is provided in order to allow the   *
1240  ! operator to make the selection of output and input desired.  Output      *
1245  ! data are available in both tabular and graphical format.  After a       *
1250  ! selection is made, the beam pattern is computed then a graph (if        *
1255  ! requested), is display for 2 secondes, then another Menu appears       *
1260  ! showing different format & medium on which it can be transfered (such   *
1265  ! as Thinkjet, Printer, Plotter, using Expanded mode or not, etc.).       *
1270  ! Afterward, a new graph is made (after a CONTINUE is press), and so on.  *
1275  !                                                                          *
1280  ! 4.     A menu is used to do individual changes of the complex sensiti-   *
1285  ! vities of the staves or the hydrophones.  The hydrophones can be padded*
```

```
1290  ! or disconnected or any value of magnitude and phase can be entered.    *
1295  !                                                                         *
1300  ! 5.     The user must ensure that a plotter is indeed available (with   *
1305  ! paper in place) when plotting is required, and that a printer is also  *
1310  ! available when printout is required.  Make sure that they are turn on. *
1315  !                                                                         *
1320  ! 5.     Refer to the User Manual and the requirement and design specifi-*
1325  ! cations of the thesis.                                                  *
1330  !                                                                         *
1335  ! *************************************************************************
1340  !                          MAIN PROGRAM
1345  ! *************************************************************************
1350  !
1355    GOSUB Define_database       ! DATABASE DEFINITION
1360    GOSUB Initialize_data       ! INITIALIZE THE DATABASE
1365    GOSUB Print_title           ! PRINT TITLE OF THE PROGRAM
1370    GOSUB Set_date_time         ! SET COMPUTER'S CLOCK IF NOT ALREADY SET
1375    GOSUB User_menu             ! ALLOW USER TO SET THE SIMULATION
1380    GOSUB Define_hydr_psn       ! DEFINE THE POSITION X, Y, & Z OF STAVES
1385                                ! AND HYDROPHONES
1390    GOSUB Array_defn            ! DEFINE TABLE OF MRAs VS SUB-ARRAYs DEFN.
1395    GOSUB Array_configure       ! CONFIGURE THE SUB-ARRAY FOR DFT OF BP
1400    GOSUB Ind_staves_menu       ! MENU TO CHANGE INDIVIDUAL STAVES STATUS
1405    GOSUB Beam_pattern          ! COMPUTE BEAM PATTERN
1410    GOSUB Display_input         ! PRODUCE A TABLE OF INPUT
1415    GOSUB Display_output        ! OUTPUT THE BEAM PATTERN COEFFICIENTS
1420    GOSUB Graph_output          ! PLOT ANY GRAPH REQUESTED IN MAIN MENU
1425    GOSUB Terminate             ! TERMINATE EXECUTION OF THIS PROGRAM
1430  !
1435  ! *************************************************************************
1440  !                    END OF MAIN ROUTINE                                  *
1445  ! *************************************************************************
1450   !
1455   ! ************************************************************************
1460 Define_database:   !  DEFINE THE DATABASE OF THE SYSTEM                    *
1465   ! ************************************************************************
1470   !                                                                        *
1475   ! MODULE NAME: DEFINE_DATABASE                DATE: 20 SEPTEMBER 1985  *
1480   !                                                                        *
1485   ! PROGRAMMER: Sylvain Fleurant                DATE: 20 SEPTEMBER 1985  *
1490   !                                                                        *
1495   ! PURPOSE: Define the database to be used by the program.                *
1500   !                                                                        *
1505   ! INPUT  : NIL                                                           *
1510   ! OUTPUT : ALL MAIN DATA ELEMENTS                                        *
1515   ! CALLS  : NIL                                                          *
1520   ! SIDE EFFECTS : NIL                                                    *
1525   ! REMARKS : NIL                                                         *
1530   ! ************************************************************************
1535   ! DATABASE
1540    OPTION BASE 1
1545                                  !
1550    DIM Actual_amp_wg(1:52,-1:1)  ! Actual amplitude weights measured.
1555    DIM Theory_amp_wg(1:52,-1:1)  ! Theoric amplitude weights if all elements
1560                                  ! are identical i.e. 1.0.
1565    DIM Actual_pha_wg(1:52,-1:1)  ! Actual phase weights measured (in degree)
1570    DIM Theory_pha_wg(1:52,-1:1)  ! Theoric phase weights if all elements are
1575                                  ! identical i.e. 0 degree.
1580                                  !
1585    DIM Time_delay(1:52)          ! Time delay between staves. It can be
```

```
1590                                ! expanded to a M x N x P matrix for single
1595                                ! elements consideration.
1600    ! ---------------------------------------------------------------------
1605                                !
1610                                ! *** ALL POSITIONS ARE IN METERS ***
1615                                !
1620    DIM X_psn(1:SZ)             ! Location of a stave along X-axis
1625    DIM Y_psn(1:SZ)             ! Location of a stave along Y-axis
1630                                ! Off center.  If the single elements are
1635                                ! at different psn, use  Y_psn(1:SZ,-1:1)
1640    DIM Z_psn(-1:1)             ! Location of an element in the vertical
1645                                ! If it is different for all element then
1650                                ! use Z_psn(1:SZ,1:SZ,-1:1)
1655                                !
1660    ! ---------------------------------------------------------------------
1665                                !
1670    DIM Actual_re_bp(-180:180)  ! Real part of the beam pattern - ACTUAL
1675    DIM Actual_im_bp(-180:180)  ! Imaginary part of the bp      - ACTUAL
1680                                !
1685    DIM Actual_mag_bp(-180:180) ! Normalized magnitude of the bp - ACTUAL
1690    DIM Actual_phase_bp(-180:180)! Phase of the beam pattern     - ACTUAL
1695                                !
1700    DIM Actual_db_bp(-180:180)  ! Mag. of the normalized bp in dB -ACTUAL
1705                                !
1710    DIM Actual_db_xaxis(-180:180)! X-axis part of the bp for the polar plot
1715    DIM Actual_db_yaxis(-180:180)! Y-axis part of the bp for the polar plot
1720                                !
1725    ! ---------------------------------------------------------------------
1730    DIM Theory_re_bp(-180:180)  ! Real part of the beam pattern - THEORY
1735    DIM Theory_im_bp(-180:180)  ! Imaginary part of the bp      - THEORY
1740                                !
1745    DIM Theory_mag_bp(-180:180) ! Normalized magnitude of the bp - THEORY
1750    DIM Theory_phase_bp(-180:180)! Phase of the beam pattern     - THEORY
1755                                !
1760    DIM Theory_db_bp(-180:180)  ! Mag. of the normalized bp in dB -THEORY
1765                                !
1770    DIM Theory_db_xaxis(-180:180)! X-axis part of the bp for the polar plot
1775    DIM Theory_db_yaxis(-180:180)! Y-axis part of the bp for the polar plot
1780                                !
1785    ! ---------------------------------------------------------------------
1790                                !
1795    DIM Um(-180:180)            ! DIRECTION COSINE Fx * LAMBDA
1800    DIM Psi(-180:180)           ! HORIZONTAL ANGLE
1805                                !
1810    ! ---------------------------------------------------------------------
1815                                !
1820    DIM Clear$[2]               ! To allow to clear whatever is on the CRT
1825    DIM Home$[2]                ! To send cursor to lower left corner
1830    DIM Medium$[5]              ! MEDIUM in which the array is positionned
1835                                ! i.e. WATER or AIR
1840    DIM Mode$[4]                ! Mode of operation - window
1845                                ! Choice = SUM ,DIFF, & BOTH
1850                                !
1855    DIM Left_activate$[5]       ! Flag used for unused portion
1860                                ! FALSE = Deactivated, TRUE = Activated
1865                                !
1870    ! ---------------------------------------------------------------------
1875                                !
1880    DIM Time_del_buffer(1:28)   ! Buffer used to read 28 time delay values
1885                                ! from data lists by the routine
```

```
1890                                ! Define_time_del.
1895                                !
1900    DIM Temp_td_buffer(1:28)    ! Same as Time_del_buffer for manipulation
1905                                !
1910    ! ------------------------------------------------------------------------
1915                                !
1920    DIM Amp_wg_buffer(1:52)     ! Buffer used to read actual complex
1925                                ! relative sensitivitiy of staves from disk
1930    DIM Pha_wg_buffer(1:52)     ! Same for the phase
1935                                !
1940    ! ------------------------------------------------------------------------
1945    !
1950    !                       MRA LOOK-UP TABLE
1955    !
1960    DIM Mra(-38:38)             ! Array of MRA in degrees from -170 to 170
1965                                !
1970                                ! The next four arrays indicates the values
1975                                ! of the element position of the first and
1980                                ! second left and right sub-arrays for a
1985                                ! specific MRA.
1990                                !
1995    DIM Left_1st_array(-38:38,1:2)   ! First left sub-array.
2000                                     ! 1 : Lower Left, 2 : Upper Left
2005    DIM Left_2nd_array(-38:38,1:2)   ! Second left sub-array.
2010    DIM Right_1st_array(-38:38,1:2)  ! First right sub-array.
2015                                     ! 1 : Lower Right, 2 : Upper Right
2020    DIM Right_2nd_array(-38:38,1:2)  ! Second right sub-array.
2025                                     !
2030    DIM Flag_2nd_left$(-38:38)[1]    ! Indicate if a second left sub-array
2035                                     ! is used. Y = Yes or N = No
2040    DIM Flag_2nd_right$(-38:38)[1]   ! Indicate if a second right sub-array
2045                                     ! is used. Y = Yes or N = No
2050                                !
2055    ! ------------------------------------------------------------------------
2060                                !
2065    DIM Stave_menu_flag$[5]     ! FLAG used by the Ind_stave_menu to
2070                                ! the user to change any complex weights
2075                                ! TRUE  = User wants to implement changes
2080                                ! FALSE = User does not want any changes
2085                                !
2090    DIM Stay_in_menu$[5]        ! FLAG to indicate weither more staves have
2095                                ! to be change or not
2100                                ! TRUE  = More changes
2105                                ! FALSE = No more to be done - exit
2110                                !
2115    ! ------------------------------------------------------------------------
2120                                !
2125                                ! for the Graph_xx_flag$ , a "YES" indicate
2130                                ! that this graph will be produce, a "NO"
2135                                ! will deny it - see MENU.
2140                                !
2145    DIM Graph_db_flag$[3]       ! FLAG to indicate if a graph of the mag.
2150                                ! in dB of the beam pattern is required.
2155    DIM Graph_ma_flag$[3]       ! FLAG to indicate if a graph of the
2160                                ! normalized beam pattern is required
2165    DIM Graph_po_flag$[3]       ! FLAG to indicate if a polar graph of the
2170                                ! beam pattern in dB is require.
2175                                !
2180    DIM Graph_op_medium$[4]     ! To indicate the type of output medium
2185                                ! Definition:  CRT_ = CRT Output
```

```
2190                                ! PRHN = Printer HP2671G Normal size
2195                                ! PRTN = Printer Thinkjet Normal size
2200                                ! PRHE = Printer HP2671G Expanded size
2205                                ! PRTE = Printer Thinkjet Expanded size
2210                                ! PLOT = Plotter Normal size
2215                                !
2220    ! ------------------------------------------------------------------------
2225                                !
2230                                !
2235    DIM Output_in_data$[3]      ! To indicate if the input data (Amplitude
2240                                ! and Phase) have to be outputed: YES or NO
2245                                ! Also named hydrophone characteristics.
2250                                !
2255    DIM Output_bp_data$[3]      ! To indicate if the computed beam pattern
2260                                ! data have to be outputed : YES or NO.
2265                                !
2270    DIM Do_plot$[5]             ! To indicate if a plot is required or not
2275                                ! TRUE = yes do one, FALSE = no plot
2280                                !
2285    DIM Lp_input_flag$[1]       ! To indicate if the output is on
2290                                ! the printer (used by Display_input
2295                                ! & Display_output).
2300                                !
2305    ! ------------------------------------------------------------------------
2310                                !
2315    DIM Read_cs_flag$[1]        ! Read the complex sensitivity from disk
2320                                ! flag, Y = Read them
2325                                !      N = Use the default value
2330                                !
2335    ! ------------------------------------------------------------------------
2340                                !
2345    DIM Temporary_char$[1]      ! Temporary char variable used in User_menu
2350    DIM Insert_line$[160]       ! Allow to input anything from the keyboard
2355    DIM Ok_input_flag$[5]       ! Indicate if the input is correct or not
2360                                ! from done from the keyboard.
2365                                !
2370    ! ------------------------------------------------------------------------
2375                                !
2380    DIM Plotter_flag$[3]        ! To indicate the presence of a plotter
2385    DIM Printer_flag$[3]        ! To indicate the presence of a printer
2390                                !
2395    ! ------------------------------------------------------------------------
2400                                  !
2405    INTEGER I,J,K,M,P             ! Used for indexing
2410                                  !
2415    INTEGER Max_angle             ! Value of max (absolute) angle in deg.
2420                                  !
2425    INTEGER Max_nn_staves         ! Actual maximum number of staves
2430    INTEGER Ttl_nn_element(-38:38)  ! Total number of elements in sub-array
2435    INTEGER Nn_left_element(-38:38) ! Number of elements in the left sub-a.
2440    INTEGER Nn_rght_element(-38:38) ! Number of elements in the right sub-a.
2445                                  !
2450    INTEGER Mra_index             ! Index pointing to the information
2455                                  ! relevant to a specific MRA.
2460                                  !
2465    INTEGER Line_type_theo        ! Line type used for theoretical or
2470                                  ! design plots of beam pattern
2475    INTEGER Line_type_actl        ! Line for the actual or degraded
2480                                  ! plots of beam patterN
2485                                  !
```

```
2490    INTEGER Screen                  ! Screen width
2495    INTEGER Center                  ! Center of the screen
2500                                    !
2505    ! ---------------------------------------------------------------------
2510                    ! VARIABLES USE BY INDIVIDUAL STAVES MENU
2515                                    !
2520    INTEGER Stave_nn_select         ! Identify the stave number presently
2525                                    ! used by the user in the Ind_staves_menu
2530                                    ! It varies from 0 to Max_nn_staves
2535                                    !
2540    INTEGER Hydro_nn                ! Hydrophone identification number.
2545                                    ! The values are -1 for A, 0 for B, +1 for
2550                                    ! C , and 3 for all three hydrophones.
2555                                    !
2560    INTEGER Padding_level           ! Indicate how many padding is done on a
2565                                    ! i.e. how many capacitors are added.
2570                                    ! Where 0 means no padding, 1 means one,...
2575                                    !
2580    INTEGER Temp_stave              ! Temporary stave number value used for
2585                                    ! input validation before assignment to
2590                                    ! to Stave_nn_select.
2595                                    !
2600    ! ---------------------------------------------------------------------
2605                                    !
2610    INTEGER From_angle_1            ! First FROM angle for polar plot (degrees)
2615    INTEGER To_angle_1              ! First TO angle for polar plot (degrees)
2620    INTEGER From_angle_2            ! Second FROM angle for polar plot
2625    INTEGER To_angle_2              ! Second TO angle for polar plot
2630                                    !
2635    ! ---------------------------------------------------------------------
2640                                    !
2645    REAL From_angle_1_r             ! Same as integer one but used for sub-
2650                                    ! program call (only return a real).
2655    REAL To_angle_1_r               ! Real counterpart of integer
2660    REAL From_angle_2_r             ! Same
2665    REAL To_angle_2_r               ! Same
2670                                    !
2675    ! ---------------------------------------------------------------------
2680                                    !
2685    REAL Graph_scale                ! Scale factor for the size of the graphs
2690                                    ! Graph_scale = 0 to 1.0 (Max size)
2695    REAL Freq                       ! Frequency
2700    REAL Lambda                     ! Wavelength
2705    REAL Minimum_nn_db              ! Minimum number of dB of the beam pattern
2710    REAL Minimum_nn_mag             ! Minimum magnitude corresponding to
2715                                    ! Minimum_nn_db
2720    REAL Normal_factor              ! Normalization factor
2725    REAL Requested_mra              ! MRA as demanded by the operator in degree
2730    REAL Used_mra                   ! Actual MRA used by the system
2735    REAL Speed                      ! Speed of sound in the medium
2740                                    !
2745    REAL Temp_mag                   ! Use to temporarily store the value of the
2750                                    ! magnitude of sensitivity of an element
2755    REAL Temp_phase                 ! Use to temporarily store the value of the
2760                                    ! phase of sensitivity of an element
2765    REAL Temporary_real             ! Temporary real variable used in User_menu
2770                                    !
2775    REAL T1                         ! Starting time of the DFT of both BP
2780    REAL T2                         ! Ending time of the DFT of both BP
2785    !
```

```
2790    RETURN
2795    ! -------------------------end-of-sub-routine-------------------------
2800    !
2805    ! ***************************************************************************
2810 Print_title:              ! PRINT THE HEADER                                    *
2815    ! ***************************************************************************
2820    ! MODULE NAME: Print_title                              DATE: 11 JUNE 1985    *
2825    !                                                                             *
2830    ! PROGRAMMER: Sylvain Fleurant                          DATE: 11 JUNE 1985    *
2835    !                                                                             *
2840    ! PURPOSE : Print the title block and necessary information to the            *
2845    !           user.                                                             *
2850    !                                                                             *
2855    ! INPUT  : NIL                                                               *
2860    ! OUTPUT : Title block.                                                      *
2865    ! CALLS  : NIL                                                              *
2870    ! SIDE EFFECTS : Clear the screen and output a title page.                   *
2875    ! REMARKS : NIL                                                             *
2880    ! ***************************************************************************
2885    !
2890      GRAPHICS OFF
2895      ALPHA ON
2900      CONTROL 1,12;1          ! Deactivate any softkeys
2905      PRINT USING "@"
2910      OUTPUT 2;Home$;         ! Clear the whole screen
2915      PRINT
2920      PRINT
2925      PRINT TAB(Center/2);"     SOFTWARE PROGRAM TO COMPUTE THE"
2930      PRINT
2935      PRINT TAB(Center/2);"       HORIZONTAL BEAM PATTERN OF A"
2940      PRINT
2945      PRINT TAB(Center/2);"HULL MOUNTED CONFORMAL ARRAY OF HYDROPHONES"
2950      PRINT USING "//"
2955      PRINT TAB(Center/2);"U.S. NAVAL POSTGRADUATE SCHOOL - MONTEREY, CA"
2960      PRINT TAB(Center/2);"                OCTOBER 1985"
2965      PRINT
2970      PRINT TAB(Center);"          by"
2975      PRINT TAB(Center);"Capt Sylvain FLEURANT"
2980      PRINT TAB(Center);"Canadian Armed Forces"
2985      PRINT
2990      PRINT
2995      PRINT "    *** Press CONTINUE to execute the program ***"
3000      PAUSE  ! Wait until the user is ready to proceed
3005      !
3010    RETURN   !END OF PRINT TITLE
3015    ! -------------------------end-of-sub-routine-------------------------
3020    !
3025    ! ***************************************************************************
3030 Initialize_data:   !   INITIALIZE THE DATABASE OF THE SYSTEM                     *
3035    ! ***************************************************************************
3040    !                                                                             *
3045    ! MODULE NAME: Initialize_data                        DATE: 23 MAY 1985       *
3050    !                                                                             *
3055    ! PROGRAMMER: Sylvain Fleurant                        DATE: 31 OCTOBER 1985   *
3060    !                                                                             *
3065    ! PURPOSE : Initialize all data of the database to preset value               *
3070    !           acording to the situation.                                        *
3075    !                                                                             *
3080    ! INPUT  : See below                                                         *
3085    ! OUTPUT : See below                                                         *
```

```
3090   ! CALLS  : NIL                                                          *
3095   ! SIDE EFFECTS : Set many variables to default values                   *
3100   ! REMARKS : NIL                                                         *
3105   ! ************************************************************************
3110   !
3115      PRINTER IS 1                    ! 1 for SCREEN - 701 for external
3120      GRAPHICS OFF
3125      ALPHA ON
3130      STATUS 1,9;Screen               ! Get the screen width
3135      Center=(Screen-36)/2            ! Center of the screen.
3140      Clear$=CHR$(255)&CHR$(75)       ! = (SHIFT)( CLR SCR) key on KBD
3145      Home$=CHR$(255)&CHR$(84)        ! = (SHIFT)(down arrow) key on
3150                                      ! KBD i.e. the HOME key
3155                                      !
3160      ! ---------------------------------------------------------------
3165      Max_angle=180                   ! Psi varies from -180 to 180 degrees
3170                                      !
3175      Normal_factor=1                 ! Set the normalization factor.
3180      Freq=1000                       ! In Hertz
3185      Speed=1500                      ! In  m/s
3190      Lambda=Speed/Freq               ! In  meters
3195                                      !
3200      Requested_mra=0.                ! In degrees
3205      Used_mra=0.                     ! In degrees
3210      Mra_index=0
3215      Minimum_nn_db=-40               ! -40 dB is the MINIMUM allowed
3220                                      !
3225      Max_nn_staves=52                ! 52 Staves in SSBN 640
3230      Ttl_nn_elements=28              ! Default is 28 elements
3235                                      !
3240      Line_type_actl=1                ! Line for the actual graph
3245      Line_type_theo=4                ! Line type used for theoretical graph
3250                                      !
3255      Graph_scale=1.00                ! Default: 1.00 Max size
3260                                      !
3265      From_angle_1=-Max_angle         ! Default for polar plot
3270      To_angle_1=Max_angle
3275      From_angle_2=0                  ! Default no second plot
3280      To_angle_2=0
3285                                      !
3290      ! ---------------------------------------------------------------
3295                                      !
3300      Medium$="WATER"
3305      Mode$="SUM "
3310      Graph_op_medium$="CRT_"         ! To indicate the use of the CRT
3315                                      !
3320      Left_activated$="FALSE"         ! Second left sub-array deactivated
3325                                      !
3330      Ok_input_flag$="FALSE"          ! Data from the keyboard are incorrect
3335                                      !
3340      Do_plot$="FALSE"
3345      !
3350      Stave_menu_flag$="FALSE"        ! User does not want to change the
3355                                      ! hydrophones' complex sensitivity
3360      !
3365      Graph_db_flag$="NO "
3370      Graph_ma_flag$="NO "
3375      Graph_po_flag$="YES"            ! Only the beam pattern in dB is done
3380      !
3385      Output_in_data$="NO "           ! NO output of the hydrophone chars.
```

```
3390      Output_bp_data$="NO "          ! NO output of the computed B. Pattern
3395      !
3400      Lp_input_flag$="N"             ! No output on the printer.
3405      !
3410      Printer_flag$="YES"            ! Usually printer is ON
3415      Plotter_flag$="NO "            ! Usually there is no plotter
3420                                     !
3425      ! ---------------------------------------------------------------------
3430                                     !
3435      RAD                            ! All angles are in radians, except
3440                                     ! for the graphs.
3445      !
3450      CONTROL 2,1;0                  ! Set PRINTALL to "OFF" (0)
3455      CONTROL 1,4;0                  ! DISPLAY FUNCTIONS to "OFF" (0)
3460      !
3465      ! REMOVE THE KEY CONTROLS AT THE BOTTOM OF THE SCREEN
3470      !
3475      CONTROL 1,12;1       ! ;1 for OFF,  0; for ON, and
3480                           ! ;2 for ON when the program is not running
3485      !
3490      ! ---------------------------------------------------------------------
3495      !
3500      !  ALL MATRIX ARE AUTOMATICALLY INITIALIZE TO ZERO BY THE SYSTEM
3505      !  NO PRE-INITIALIZATION IS REQUIRED
3510      !
3515    RETURN
3520  ! -------------------------end-of-sub-routine---------------------------
3525  !
3530  ! ***********************************************************************
3535 Define_hydr_psn: ! DEFINE THE POSITION X, Y, & Z OF STAVES AND HYDROPHONES*
3540  ! ***********************************************************************
3545  ! MODULE DEFINE HYDROHONES POSITION    VERSION 1.0  23 SEPTEMBER 1985  *
3550  !                                                                      *
3555  ! PROGRAMMER: SYLVAIN FLEURANT                DATE : 23 SEPTEMBER 1985  *
3560  !                                                                      *
3565  ! PURPOSE: To define the look-up table for the actual location  of the  *
3570  !          staves and single elements i.e. x, y, and z in meters.      *
3575  !                                                                      *
3580  ! INPUT:   X_psn, Y_psn, and Z_psn.                                    *
3585  ! OUTPUT:  X_psn, Y_psn, and Z_psn.                                    *
3590  ! SIDE EFFECTS: NIL                                                    *
3595  ! REMARKS: ALL THE DATA ARE IN METERS FOR A SPECIFIC SUBMARINE.        *
3600  !          SSBN 640 (7)                                              *
3605  !***********************************************************************
3610    !
3615    ! DEFINE LOCAL DATA BASE
3620    !
3625    REAL Offset              ! Offset in meters used to adjust old to new
3630                             ! coordinates system
3635    DISP "HYDROPHONE POSITIONS BEING COMPUTED"
3640    ! ---------------------------------------------------------------------
3645    !
3650    ! DEFINE Z POSITION IN METERS
3655    !
3660      Z_psn(1)=-.762         ! 76.2 CM OR 30 INCHES
3665      Z_psn(0)=0.
3670      Z_psn(-1)=+.762
3675    !
3680    ! ---------------------------------------------------------------------
3685    !
```

```
3690     ! DEFINE X POSITION IN METERS
3695     !
3700     ! (1) 48' 19/32"  , (2) 46' 13/16"  , (3) 44' 1-1/16", (4) 42' 4-11/32"
3705     DATA  14.645481 , 14.041437 , 13.438187 , 12.911931      ! meters
3710     ! (5) 40' 1-9/16" , (6) 38' 1-13/16", (7) 36' 2-1/8" , (8) 34' 6-1/32"
3715     DATA  12.231687 , 11.628437 , 11.026774 , 10.516393      ! meters
3720     ! (9) 32' 2-11/16", (10) 30' 3-1/16", (11) 28' 3-1/2", (12) 26' 3-7/8"
3725     DATA  9.8523423 , 9.2217873 , 8.6232997 , 8.0232247      ! meters
3730     !(13) 24' 4-3/8", (14) 22' 4-15/16", (15) 20' 9-23/32",(16) 18' 1-9/16"
3735     DATA  7.4263248 , 6.8310123 , 6.3428559 , 5.5260873      ! meters
3740     !(17) 16' 7-3/16", (18) 14' 8-1/4", (19) 12' 9-1/2" , (20) 10' 11-3/16"
3745     DATA  5.0593623 , 4.47675 , 3.8988997 , 3.3321622        ! meters
3750     !(21) 9' 1-7/32  ", (22) 7' 2-7/16" , (23) 5' 7-1/2" , (24) 4' 0-15/16"
3755     DATA  2.7741562 , 2.1955125 , 1.7145 , 1.2430125         ! meters
3760     !(25) 2' 7"       , (26) 1' 6-1/4"
3765     DATA  0.7873999 , 0.4635499                              ! meters
3770     !
3775     FOR I=Max_nn_staves TO (Max_nn_staves/2+1) STEP -1
3780           READ X_psn(I)
3785           X_psn(Max_nn_staves-I+1)=X_psn(I)
3790     NEXT I
3795     !
3800     Offset=X_psn(1)/2
3805     MAT X_psn= (Offset)-X_psn
3810     !
3815     ! ---------------------------------------------------------------------
3820     !
3825     ! DEFINE Y POSITION IN METERS    (Numbers in parenthesis are staves #)
3830     !
3835     ! (1) 14' 4-1/2"  , (2) 14' 2-1/2"   , (3) 14' 0-3/8" , (4) 13' 11-1/2"
3840     DATA  4.381500 , 4.3306998 , 4.276725 , 4.2544998
3845     ! (5) 13' 7-11/16", (6) 13' 5-1/8"   , (7) 13' 2-1/4" , (8) 13' 0-1/32"
3850     DATA  4.1576625 , 4.0925748 , 4.019550 , 3.9631936       ! meters
3855     ! (9) 12' 8-3/16" , (10) 12' 4-13/16", (11) 12' 1-1/4", (12) 11' 9-1/2"
3860     DATA  3.8655622 , 3.7798372 , 3.6890449 , 3.5940997      ! meters
3865     !(13) 11' 5-11/32", (14) 11' 1-1/16" , (15) 10' 8-9/16",(16) 10' 15/16"
3870     DATA  3.488531 , 3.3797872 , 3.2654872 , 3.0718125       ! meters
3875     !(17)  9' 8-15/16", (18) 9' 2-7/16" , (19) 8' 7-1/8", (20) 7' 10-13/16"
3880     DATA   2.9702124 , 2.8051127 , 2.619375 , 2.4082374      ! meters
3885     !(21)  7' 1-1/2"  , (22) 6' 3-1/4"  , (23) 5' 3-5/16", (24) 4' 0-1/4"
3890     DATA   2.1717 , 1.9113499 , 1.6081374 , 1.2255499        ! meters
3895     !(25)  2' 8"       , (26) 1' 0"
3900     DATA   0.817999 , 0.3048                                 ! meters
3905     !
3910      FOR I=Max_nn_staves TO (Max_nn_staves/2+1) STEP -1
3915            READ Y_psn(I)
3920            Y_psn(Max_nn_staves-I+1)=-Y_psn(I)
3925      NEXT I
3930     !
3935     ! ---------------------------------------------------------------------
3940     !
3945     !  PRINT THE DATA POINT - DEACTIVATE AT THE PRESENT TIME
3950     !
3955     !PRINT
3960     !PRINT " STAVE ##    X POSITION    Y POSITION"
3965     !PRINT "               meters         meters"
3970     !PRINT
3975     !FOR J=1 TO Max_nn_staves
3980     !      PRINT USING Form_elemen_psn;J,X_psn(J),Y_psn(J)
3985     !NEXT J
```

```
3990 Form_elemen_psn:   IMAGE 2X,DD,8X,3D.5D,6X,3D.5D
3995     !
4000     DISP " "
4005   RETURN  !DEFINE_HYDRO_PSN
4010   ! -------------------------end-of-sub-routine--------------------------
4015   !
4020   ! ***************************************************************************
4025 Array_defn:  ! ARRAY DEFINITION PER MAXIMUM RESPONSE ANGLE                    *
4030   ! ***************************************************************************
4035   !                                                                           *
5905   ! MODULE NAME : ARRAY DEFINITION              DATE: 25 SEPTEMBER 1985     *
4040   !                                                                           *
4045   ! PURPOSE: Initialize the look up table use to define the first and        *
4050   !          second left and right sub-arrays used for a given Maximum      *
4055   !          Response Angle (in degrees).                                   *
4060   !                                                                           *
4065   ! INPUT: DATA, MRA, Left_1st_array, Left_2nd_array, Right_1st_array,      *
4070   !        Right_2nd_array, Flag_2nd_left$, Flag_2nd_right$,                *
4075   !        Ttl_nn_element, Nn_rght_element, Nn_left_element, &              *
4080   !        Nn_mra_data_pt.                                                  *
4085   ! OUTPUT: Same as the INPUT (except DATA).                                *
4090   ! SIDE EFFECTS: NIL                                                       *
4095   ! REMARKS: NIL                                                           *
4100   ! ***************************************************************************
4105     !
4110     ! DEFINE LOCAL DATA BASE
4115     !
4120     DISP "SUB-ARRAYS BEING COMPUTED"
4125     INTEGER Nn_mra_data_pt    ! Number of data points to define the
4130                               ! Look-up table of MRA. (TOTAL - 1)
4135                               !
4140     Nn_mra_data_pt=38         ! FROM 0 TO 38 - IMPORTANT DATA VALUE
4145                               ! EQUAL TO THE TOTAL NUMBER OF DATA
4150                               ! POINTS WRITTEN IN THIS MODULE
4155     ! ---------------------------------------------------------------------
4160     !
4165     ! DEFINE THE DATA BASE OF SUB-ARRAYS
4170     !
4175     ! MRA, TTL_NN_ELE,#LEFT,FLAG_LEFT,L1AR,L2AR,#RIGHT,FLAG_RIGHT,R1AR,R2AR
4180     DATA 000  , 28 , 14, "N", 13, 26,  0,  0, 14, "N", 27, 40,  0,  0
4185     DATA 005  , 26 , 12, "N", 16, 27,  0,  0, 14, "N", 28, 41,  0,  0
4190     DATA 010  , 26 , 12, "N", 18, 29,  0,  0, 14, "N", 30, 43,  0,  0
4195     DATA 015  , 25 , 11, "N", 20, 30,  0,  0, 14, "N", 31, 44,  0,  0
4200     DATA 020  , 26 , 12, "N", 21, 32,  0,  0, 14, "N", 33, 46,  0,  0
4205     DATA 025  , 26 , 12, "N", 22, 33,  0,  0, 14, "N", 34, 47,  0,  0
4210     DATA 030  , 26 , 12, "N", 23, 34,  0,  0, 14, "N", 35, 48,  0,  0
4215     DATA 035  , 26 , 12, "N", 23, 34,  0,  0, 14, "N", 35, 48,  0,  0
4220     DATA 040  , 26 , 12, "N", 24, 35,  0,  0, 14, "N", 36, 49,  0,  0
4225     DATA 045  , 27 , 13, "N", 24, 36,  0,  0, 14, "N", 37, 50,  0,  0
4230     DATA 050  , 27 , 13, "N", 25, 37,  0,  0, 14, "N", 38, 51,  0,  0
4235     DATA 055  , 28 , 14, "N", 25, 38,  0,  0, 14, "N", 39, 52,  0,  0
4240     DATA 060  , 28 , 14, "N", 25, 38,  0,  0, 14, "N", 39, 52,  0,  0
4245     DATA 065  , 28 , 14, "N", 25, 38,  0,  0, 14, "N", 39, 52,  0,  0
4250     DATA 070  , 27 , 13, "N", 26, 38,  0,  0, 14, "N", 39, 52,  0,  0
4255     DATA 075  , 27 , 14, "N", 26, 39,  0,  0, 13, "N", 40, 52,  0,  0
4260     DATA 080  , 26 , 13, "N", 27, 39,  0,  0, 13, "N", 40, 52,  0,  0
4265     DATA 085  , 26 , 13, "N", 27, 39,  0,  0, 13, "N", 40, 52,  0,  0
4270     DATA 090  , 26 , 13, "N", 27, 39,  0,  0, 13, "N", 40, 52,  0,  0
4275     DATA 095  , 26 , 14, "N", 27, 40,  0,  0, 12, "N", 41, 52,  0,  0
4280     DATA 100  , 25 , 13, "N", 28, 40,  0,  0, 12, "N", 41, 52,  0,  0
```

```
4285    DATA 105  , 25 , 13, "N", 28, 40,  0,  0, 12, "N", 41, 52,  0,  0
4290    DATA 110  , 25 , 13, "N", 28, 40,  0,  0, 12, "N", 41, 52,  0,  0
4295    DATA 115  , 25 , 13, "N", 28, 40,  0,  0, 12, "N", 41, 52,  0,  0
4300    DATA 120  , 25 , 14, "N", 28, 41,  0,  0, 11, "N", 42, 52,  0,  0
4305    DATA 125  , 24 , 13, "N", 29, 41,  0,  0, 11, "N", 42, 52,  0,  0
4310    DATA 130  , 23 , 12, "N", 30, 41,  0,  0, 11, "N", 42, 52,  0,  0
4315    DATA 135  , 22 , 12, "N", 31, 42,  0,  0, 10, "N", 43, 52,  0,  0
4320    DATA 140  , 21 , 11, "N", 32, 42,  0,  0, 10, "N", 43, 52,  0,  0
4325    DATA 145  , 19 , 10, "N", 34, 43,  0,  0,  9, "N", 44, 52,  0,  0
4330    DATA 150  , 18 , 10, "N", 35, 44,  0,  0,  8, "N", 45, 52,  0,  0
4335    DATA 155  , 18 , 10, "N", 35, 44,  0,  0,  8, "N", 45, 52,  0,  0
4340    DATA 160  , 17 ,  9, "N", 36, 44,  0,  0,  8, "N", 45, 52,  0,  0
4345    DATA 165  , 17 ,  9, "N", 37, 45,  0,  0,  8, "Y", 46, 52,  1,  0
4350    DATA 166  , 16 ,  8, "N", 38, 45,  0,  0,  8, "Y", 46, 52,  1,  0
4355    DATA 167  , 16 ,  8, "N", 40, 47,  0,  0,  8, "Y", 48, 52,  1,  3
4360    DATA 168  , 16 ,  8, "N", 41, 48,  0,  0,  8, "Y", 49, 52,  1,  4
4365    DATA 169  , 16 ,  8, "N", 43, 50,  0,  0,  8, "Y", 51, 52,  1,  6
4370    DATA 170  , 16 ,  8, "N", 43, 52,  0,  0,  8, "N",  1,  8,  0,  0
4375    !
4380    ! ---------------------------------------------------------------------
4385    ! READ THE DATA FROM THE DATA TABLE
4390    !
4395    FOR I=0 TO Nn_mra_data_pt STEP 1
4400         READ Mra(I)
4405         READ Ttl_nn_element(I)
4410         READ Nn_left_element(I)
4415         READ Flag_2nd_left$(I)
4420         READ Left_1st_array(I,1),Left_1st_array(I,2)
4425         READ Left_2nd_array(I,1),Left_2nd_array(I,2)
4430         READ Nn_rght_element(I)
4435         READ Flag_2nd_right$(I)
4440         READ Right_1st_array(I,1),Right_1st_array(I,2)
4445         READ Right_2nd_array(I,1),Right_2nd_array(I,2)
4450    NEXT I
4455    !
4460    ! ---------------------------------------------------------------------
4465    !
4470    !  COMPLETE THE MRA TABLE FOR NEGATIVE MRA USING THE POSITIVE MRA
4475    !
4480    FOR I=1 TO Nn_mra_data_pt STEP 1
4485       Mra(-I)=-Mra(I)
4490       Ttl_nn_element(-I)=Ttl_nn_element(I)
4495       Nn_left_element(-I)=Nn_left_element(I)
4500       Flag_2nd_left$(-I)=Flag_2nd_left$(I)
4505       Left_1st_array(-I,2)=(Max_nn_staves+1)-Left_1st_array(I,1)
4510       Left_1st_array(-I,1)=(Max_nn_staves+1)-Left_1st_array(I,2)
4515       IF Flag_2nd_left$(I)="Y" THEN
4520            IF Left_2nd_array(I,1)<>0 THEN
4525                 Left_2nd_array(-I,2)=(Max_nn_staves+1)-Left_2nd_array(I,1)
4530            END IF
4535            IF Left_2nd_array(I,2)<>0 THEN
4540                 Left_2nd_array(-I,1)=(Max_nn_staves+1)-Left_2nd_array(I,2)
4545            END IF
4550       END IF   ! Of Flag_2nd_left$
4555       Nn_rght_element(-I)=Nn_rght_element(I)
4560       Flag_2nd_right$(-I)=Flag_2nd_right$(I)
4565       Right_1st_array(-I,2)=(Max_nn_staves+1)-Right_1st_array(I,1)
4570       Right_1st_array(-I,1)=(Max_nn_staves+1)-Right_1st_array(I,2)
4575       IF Flag_2nd_right$(I)="Y" THEN
4580            IF Right_2nd_array(I,1)<>0 THEN
```

```
4585              Right_2nd_array(-I,2)=(Max_nn_staves+1)-Right_2nd_array(I,1)
4590            END IF
4595            IF Right_2nd_array(I,2)<>0 THEN
4600              Right_2nd_array(-I,1)=(Max_nn_staves+1)-Right_2nd_array(I,2)
4605            END IF
4610       END IF   ! Of Flag_2nd_left$
4615     NEXT I
4620     !
4625     ! ---------------------------------------------------------------------
4630     !
4635     ! PRINT ALL THE ARRAY DEFINITION TABLE *** DISCONNECTED SECTION ***
4640     !
4645     !FOR I=-Nn_mra_data_pt TO Nn_mra_data_pt STEP 1
4650     !     PRINT USING "///"
4655     !     PRINT "MRA              = ";Mra(I)
4660     !     PRINT "TTL_NN_ELEMENT   = ";Ttl_nn_element(I)
4665     !     PRINT
4670     !     PRINT "NN_LEFT_ELEMENT = ";Nn_left_element(I);" FROM ";Left_1st_a
rray(I,1);" TO ";Left_1st_array(I,2)
4675     !     PRINT "ANY 2ND LEFT ? ";Flag_2nd_left$(I);" FROM ";Left_2nd_array
(I,1);" TO ";Left_2nd_array(I,2)
4680     !     PRINT
4685     !     PRINT "NN_RGHT_ELEMENT = ";Nn_rght_element(I);" FROM ";Right_1st_
array(I,1);" TO ";Right_1st_array(I,2)
4690     !     PRINT "ANY 2ND RIGHT ? ";Flag_2nd_right$(I);" FROM ";Right_2nd_ar
ray(I,1);" TO ";Right_2nd_array(I,2)
4695     !NEXT I
4700     DISP " "
4705     RETURN ! of Array_defn
4710   !
4715   ! ---------------------------end-of-sub-routine---------------------------
4720   !
4725   ! ************************************************************************
4730 Set_date_time:  !          SET TIME AND DATE FOR RECORD PURPOSES           *
4735   ! ************************************************************************
4740   !                                                                        *
4745   ! MODULE NAME: Set_date_time                          DATE: 26 JUNE 1985  *
4750   !                                                                        *
4755   ! PURPOSE : This routine check if the date and time is as it should     *
4760   !           be, if not enter it                                          *
4765   !                                                                        *
4770   ! INPUT   : TIMEDATE                                                     *
4775   ! OUTPUT : TIMEDATE                                                     *
4780   ! CALLS   : NIL                                                         *
4785   ! SIDE EFFECTS : Set time & date if necessary                           *
4790   ! REMARKS : This is executed only if the computer was turn off then on  *
4795   !           without a good date & time. It is not executed if the time  *
4800   !           is correct.                                                  *
4805   !           Original programmer: Rick Self                               *
4810   ! ************************************************************************
4815   !
4820     IF TIMEDATE<2.11E+11 THEN          ! NEEDS TO BE UPDATED if default
4825                                        ! value is set for a small #
4830       DIM Today$[11]
4835       DIM Clock$[5]
4840       PRINTER IS 1
4845       PRINT USING "@"
4850       PRINT " Enter the date and time, SEPARATED by a COMMA,"
4855       PRINT " in the EXACT format shown below:"
4860       PRINT
```

```
4865          PRINT "     4 JUL 1985,9:08         "
4870          PRINT
4875          PRINT " Don't use leading zeroes EXCEPT for MINUTES ! "
4880          ON ERROR GOTO 4885
4885          BEEP 1000,.2
4890          INPUT "Date  MONTH  Year (4 digits), Hours:Minutes ",Today$,Clock$
4895          SET TIMEDATE DATE(Today$)+TIME(Clock$)
4900          DISP TIME$(TIMEDATE),DATE$(TIMEDATE)
4905          BEEP 200,.2
4910          WAIT 1                                  ! WAIT FOR 1 SECONDE
4915          Timedate=TIMEDATE                       ! Needed for READ statement
4920          OFF ERROR
4925        END IF
4930        RETURN
4935      ! ------------------------end-of-sub-routine-------------------------
4940      !
4945      !
4950      ! ***************************************************************************
4955 User_menu:    ! ALLOW USER TO USE A MENU TO SELECT THE PROPER ENVIRONMENT  *
4960      ! ***************************************************************************
4965      !                                                                             *
4970      ! MODULE NAME: User_menu                                  DATE: 20 JULY 1985  *
4975      !                                                                             *
4980      ! PURPOSE : To allow a user (i.e. an operator) to use a MENU to select  *
4985      !           the propert environment in which he wishes to do the simula-*
4990      !           lation. The user can allow multiple graph outputs, and      *
4995      !           hard copy, along with the choice of frequency, spacing      *
5000      !           between the elements, etc.                                  *
5005      !                                                                       *
5010      ! INPUT  : Output_in_data$, Output_bp_data$, Graph_ma_flag$,           *
5015      !           Graph_db_flag$, Grph_po_flag$, Speed, Lambda, Freq,         *
5020      !           Requested_mra, Mode$, Temporary_real, Temporary_char$[1]    *
5025      !           Insert_line$, Ok_input_flag$.                               *
5030      ! OUTPUT : Same as input                                                *
5035      ! CALLS  : NIL                                                         *
5040      ! SIDE EFFECTS : NIL                                                   *
5045      ! REMARKS : 1.  Many variables are modified by this program.           *
5050      !               Pay attention to them.                                  *
5055      !           2.  This sub_routine was written using an example in        *
5060      !               Chapter 10 of the HP BASIC PROGRAMMING TECHNIQUES.      *
5065      !                                                                       *
5070      ! ***************************************************************************
5075        !
5080        ! ---------------------------------------------------------------
5085        !
5090        OUTPUT 2;Clear$;          ! Clear whatever is on the screen
5095        !
5100 Menu1:       !          MAIN MENU    (i.e. First MENU )
5105        PRINT TABXY(1,1)          ! Start on top left corner.
5110        OUTPUT 2;Home$;           ! Send cursor home
5115        PRINT TAB(Center+15);"MENU"
5120        PRINT TAB(Center);"KEY       PURPOSE";TAB(Center+31);"STATUS"
5125        PRINT TAB(Center);"-----------------------------------------"
5130        PRINT
5135        PRINT TAB(Center);"K0    OUTPUT HYDRO SENSITIVITY";TAB(Center+33);Outp
ut_in_data$
5140        PRINT TAB(Center);"K1    OUTPUT BEAM PATTERN DATA";TAB(Center+33);Outp
ut_bp_data$
5145        PRINT TAB(Center);"K2    GRAPH OF MAG. IN dB";TAB(Center+33);Graph_db_
flag$
```

```
5150      PRINT TAB(Center);"K3    GRAPH OF NORM. MAG.";TAB(Center+33);Graph_ma_
flag$
5155      PRINT TAB(Center);"K4    POLAR GRAPH OF MAG.";TAB(Center+33);Graph_po_
flag$
5160      PRINT
5165      PRINT TAB(Center);"K5    OPERATING FREQUENCY";TAB(Center+33);Freq;" Hz
   . . "
5170      PRINT TAB(Center);"K6    MRA                 ";TAB(Center+33);Requested
_mra;" degrees     "
5175      PRINT TAB(Center);"K7    SPEED OF SOUND in water";TAB(Center+33);Speed
;" m/sec     "
5180      PRINT TAB(Center);"K8    MODE of OPERATION  ";TAB(Center+33);Mode$
5185      PRINT TAB(Center);"K9    STARTS PROGRAM"
5190     !PRINT
5195      PRINT "Use the softkeys below to make a selection if "
5200      PRINT "your choice is different then the default values shown"
5205      !
5210      !      All graphs can be choosen
5215      !
5220      ! ------------------------------------------------------------------
5225      !
5230       CONTROL 1,12;0     ! Activate the softkey function
5235          ON KEY 0 LABEL "OUTPUT HYDROPHONE CHARACTERISTICS" GOTO Indata_key
lbl
5240          ON KEY 1 LABEL "OUTPUT B.PATTERN" GOTO Outdata_keylbl
5245          ON KEY 2 LABEL "MAGNITUDE dB" GOTO Graph_db_keylbl
5250          ON KEY 3 LABEL "NORMALIZE MAGN." GOTO Graph_ma_keylbl
5255          ON KEY 4 LABEL "POLAR PLOT MAG." GOTO Graph_po_keylbl
5260          !
5265          ON KEY 5 LABEL "FREQUENCY" GOTO Freq_keylbl
5270          ON KEY 6 LABEL "MRA     " GOTO Mra_keylbl
5275          ON KEY 7 LABEL "SPEED of SOUND" GOTO Speed_keylbl
5280          ON KEY 8 LABEL "MODE" GOTO Menu_mode
5285          ON KEY 9 LABEL "START" GOTO Start_lbl
5290      !
5295      ! ------------------------------------------------------------------
5300      !
5305 Spin1:   GOTO Spin1       ! Wait for softkey interrupt - i.e. Operator I/O
5310      !
5315      ! ------------------------------------------------------------------
5320      !
5325 Indata_keylbl:    ! CHANGES STATUS OF THE FLAG OF OUTPUT OF INITIAL
5330                   ! HYDROPHONE CHARACTERISTICS
5335          IF Output_in_data$="YES" THEN
5340               Output_in_data$="NO "
5345          ELSE
5350               Output_in_data$="YES"
5355          END IF
5360          GOTO Menu1
5365      !
5370      ! ------------------------------------------------------------------
5375      !
5380 Outdata_keylbl:   ! CHANGES STATUS OF THE FLAG OF OUTPUT OF COMPUTED
5385                   ! BEAM PATTERN
5390          IF Output_bp_data$="YES" THEN
5395               Output_bp_data$="NO "
5400          ELSE
5405               Output_bp_data$="YES"
5410          END IF
5415          GOTO Menu1
```

```
5420      !
5425      ! ------------------------------------------------------------------
5430      !
5435 Graph_db_keylbl: ! CHANGES STATUS OF FLAG TO CONTROL GRAPH IN dB
5440          IF Graph_db_flag$="YES" THEN
5445               Graph_db_flag$="NO "
5450          ELSE
5455               Graph_db_flag$="YES"
5460          END IF
5465          GOTO Menu1
5470      !
5475      ! ------------------------------------------------------------------
5480      !
5485 Graph_ma_keylbl: ! CHANGES STATUS OF FLAG TO CONTROL GRAPH OF
5490                  ! NORMALIZED MAGNITUDE OF BEAM PATTERN
5495          IF Graph_ma_flag$="YES" THEN
5500               Graph_ma_flag$="NO "
5505          ELSE
5510               Graph_ma_flag$="YES"
5515          END IF
5520          GOTO Menu1
5525      !
5530      ! ------------------------------------------------------------------
5535      !
5540 Graph_po_keylbl: ! CHANGES STATUS OF FLAG TO CONTROL GRAPH OF
5545                  ! POLAR PLOT OF THE BEAM PATTERN
5550          IF Graph_po_flag$="YES" THEN
5555               Graph_po_flag$="NO "
5560          ELSE
5565               Graph_po_flag$="YES"
5570          END IF
5575          GOTO Menu1
5580      !
5585      ! ------------------------------------------------------------------
5590      !
5595 Freq_keylbl:     ! ALLOW CHANGES OF OPERATING FREQUENCY
5600          CONTROL 1,12;1          ! TURN OFF softkeys
5605           Insert_line$=" "       ! Initialize to nothing
5610          !
5615          ! MAKE SURE THAT ALL INPUT DATA ARE OF THE CORRECT TYPE
5620          ! This part will accept any type of input data and process it
5625          !
5630          REPEAT
5635              INPUT "ENTER YOUR FREQUENCY IN Hz",Insert_line$
5640              ! Check if the string is numeric - Pass by value
5645              Ok_input_flag$=FNIs_string_pnum$((Insert_line$))
5650              IF Ok_input_flag$="TRUE " THEN
5655                   ! This is a numeric string
5660                   ! Enter the first real number in the string
5665                   ! skip all the rest if any characters are left
5670                   ENTER Insert_line$;Temporary_real
5675                   Freq=Temporary_real
5680              ELSE
5685                   ! This is not a POSITIVE number
5690                   DISP " YOU MUST ENTER A POSITIVE NUMBER"
5695                   WAIT .75   ! Wait 3/4 sec so user can read it
5700              END IF    ! of Ok_input_flag$ is TRUE
5705              IF Freq<1.E-40 THEN ! This is probalbly 0
5710                   Freq=1000
5715                   DISP " YOU CANNOT ENTER SUCH A SMALL FREQUENCY"
```

```
5720                  WAIT .75   ! so user can read it
5725                  Ok_input_flag$="FALSE"
5730             END IF  ! of FREQ = 0
5735          UNTIL Ok_input_flag$="TRUE "    ! i.e. a good frequency
5740          Lambda=Speed/Freq               ! Recompute the wavelength
5745          CONTROL 1,12;0       ! Reactivate the softkeys
5750          GOTO Menu1
5755       !
5760       ! ------------------------------------------------------------------
5765       !
5770 Mra_keylbl:  ! ALLOW CHANGES OF THE DESIRED MRA
5775          CONTROL 1,12;1        ! TURN OFF the softkeys
5780           Insert_line$=" "     ! Initialize to nothing
5785          ! MAKE SURE THAT ALL INPUT DATA ARE OF THE CORRECT TYPE
5790          ! This part will accept any type of input data and process it
5795          !
5800          REPEAT
5805              INPUT "ENTER THE MAXIMUM RESPONSE ANGLE IN DEGREES",Insert_lin
e$
5810              ! Check if the string is numeric - Pass by value
5815              Ok_input_flag$=FNIs_string_num$((Insert_line$))
5820              IF Ok_input_flag$="TRUE " THEN
5825                    ! This is a numeric string
5830                    ! Enter the first real number in the string
5835                    ! skip all the rest if any characters are left
5840                    ENTER Insert_line$;Temporary_real
5845                    IF ((Temporary_real>180) OR (Temporary_real<-180)) THEN
5850                         DISP " YOU MUST ENTER A NUMBER BETWEEN -180 TO 180"
5855                         WAIT .75 ! Wait so the user can read it
5860                         Ok_input_flag$="FALSE"
5865                    ELSE
5870                         ENTER Insert_line$;Temporary_real
5875                         Requested_mra=Temporary_real
5880                         Ok_input_flag$="TRUE "
5885                    END IF    ! range -180 TO 180 degrees
5890              ELSE
5895                   ! This is not a POSITIVE number
5900                   DISP " YOU MUST ENTER A NUMBER"
5905                   WAIT .75   ! Wait 3/4 sec so user can read it
5910              END IF    ! of Ok_input_flag$ is TRUE
5915          UNTIL Ok_input_flag$="TRUE "    ! i.e. a good frequency
5920          !
5925          CONTROL 1,12;0     ! Reactivate the softkeys
5930          GOTO Menu1
5935       !
5940       ! ------------------------------------------------------------------
5945       !
5950 Speed_keylbl: ! ALLOW CHANGES OF THE SPEED OF SOUND IN WATER
5955          CONTROL 1,12;1        ! TURN OFF the softkeys
5960           Insert_line$=" "     ! Initialize to nothing
5965          ! MAKE SURE THAT ALL INPUT DATA ARE OF THE CORRECT TYPE
5970          REPEAT
5975              INPUT "ENTER THE SPEED OF SOUND IN METERS/SE",Insert_line$
5980              ! Check if the string is positive numeric - Pass by value
5985              Ok_input_flag$=FNIs_string_pnum$((Insert_line$))
5990              IF Ok_input_flag$="TRUE " THEN
5995                    ! This is a numeric string
6000                    ! Enter the first real number in the string
6005                    ! skip all the rest if any characters are left
6010                    ENTER Insert_line$;Temporary_real
```

```
6015                    Speed=(Temporary_real)
6020                    IF (Speed>1600) OR (Speed<1400) THEN
6025                        DISP " YOUR NUMBER IS NOT PHYSICALLY POSSIBLE"
6030                        WAIT 2.0    ! Wait 2 sec
6035                        Ok_input_flag$="FALSE"
6040                    END IF
6045               ELSE
6050                    ! This is not a POSITIVE number
6055                    DISP " YOU MUST ENTER A POSITIVE NUMBER"
6060                    WAIT .75   ! Wait 3/4 sec so user can read it
6065               END IF    ! of Ok_input_flag$ is TRUE
6070          UNTIL Ok_input_flag$="TRUE "    ! i.e. a good frequency
6075          !
6080          Lambda=Speed/Freq        ! Recompute the wavelenght
6085          CONTROL 1,12;0     ! Reactivate the softkeys
6090          GOTO Menu1
6095       !
6100       ! ---------------------------------------------------------------------
6105       !
6110 Menu_mode:           ! MENU FOR MODE OF OPERATION
6115       !
6120          IF Mode$="SUM " THEN
6125               Mode$="DIFF"
6130          ELSE
6135               Mode$="SUM "
6140          END IF
6145          GOTO Menu1
6150       !
6155       ! ---------------------------------------------------------------------
6160       !
6165 Start_lbl:       ! ENDS STATUS CHANGES SUB-ROUTINE
6170          CONTROL 1,12;1         ! Turn off the softkey function and display
6175          OUTPUT 2;Clear$;       ! Clear the screen
6180       RETURN    ! END OF USER_MENU
6185    !
6190    ! -------------------------end-of-sub-routine--------------------------
6195    !
6200    ! *************************************************************************
6205 Find_mra_index: !     SEARCH IN MRA LOOK-UP TABLE FOR MRA INDEX           *
6210    ! *************************************************************************
6215    !                                                                       *
6220    ! MODULE NAME: FIND MRA INDEX                  DATE : 29 SEPTEMBER 1985 *
6225    !                                                                       *
6230    ! PROGRAMMER: SYLVAIN FLEURANT                 DATE : 29 SEPTEMBER 1985 *
6235    !                                                                       *
6240    ! PURPOSE : To search the MRA look-up table using Used_mra for the      *
6245    !           corresponding Mra_index to be used with the table.          *
6250    !                                                                       *
6255    ! INPUT  : Used_mra, Mra, Nn_mra_data_pt                                *
6260    ! OUTPUT : Mra_index                                                    *
6265    ! SIDE EFFECTS : NIL                                                    *
6270    ! REMARKS : 1. MRA is scan from -38 TO 38 i.e. Nn_mra_data_pt in MRA    *
6275    !           Look-up table.                                              *
6280    !                                                                       *
6285    !           2. A standard binary search is done on the ordered MRA      *
6290    !           look-up table.                                              *
6295    ! *************************************************************************
6300       !
6305       ! Local Data Base
6310       !
```

```
6315     INTEGER Low                    ! Index at the bottom
6320     INTEGER High                   ! Index at the top
6325     INTEGER Middle                 ! Index in the middle, then as Mra_index
6330     !
6335     Low=-Nn_mra_data_pt
6340     High=Nn_mra_data_pt
6345     !
6350     REPEAT
6355         Middle=(Low+High) DIV 2
6360         IF (INT(Used_mra)<Mra(Middle)) THEN
6365             High=Middle-1   ! Confine search to first half of array
6370         ELSE
6375             IF (INT(Used_mra)>Mra(Middle)) THEN
6380                 Low=Middle+1  ! Confine search to second half of array
6385             END IF
6390         END IF  ! of Used_mra < Mra(Middle)
6395     UNTIL ((INT(Used_mra)=Mra(Middle)) OR (Low>High))
6400     IF Low>High THEN       ! Value was not found
6405         ! An error did occured
6410         PRINT "ERROR - ANGLE IN MRA LOOK-UP TABLE WAS NOT FOUND"
6415         GOTO Terminate
6420     ELSE
6425         Mra_index=Middle
6430     END IF  ! Low > High
6435   !
6440   RETURN
6445   !-------------------------end-of-sub-routine----------------------------
6450   !
6455   ! ****************************************************************************
6460 Array_configure: ! CONFIGURE THE SUB-ARRAYS TO BE USED FOR BP COMPUTATION *
6465   ! ****************************************************************************
6470   !                                                                            *
6475   ! MODULE NAME: ARRAY_CONFIGURE                      DATE: 20 OCTOBER 1985  *
6480   !                                                                            *
6485   ! PROGRAMMER: Sylvain Fleurant                      DATE: 20 OCTOBER 1985  *
6490   !                                                                            *
6495   ! PURPOSE : Configure the array before calculation of the beam pattern,  *
6500   !           identify exact MRA according to those available, set time    *
6505   !           delay, extract complex weights, and set the window.          *
6510   !                                                                        *
6515   ! INPUT: Requested_mra, Mode$                                            *
6520   ! OUTPUT: Used_mra, Mra_index                                            *
6525   ! SIDE EFFECTS: Extraction of complex weights by the Defin_complexwg     *
6530   !               routine may require the use of a disk drive.             *
6535   ! REMARKS: Other variables are affected by the routine calls.            *
6540   ! ****************************************************************************
6545     !
6550     !  IDENTIFY USED MRA AND INDEX OF MRA LOOK-UP TABLE
6555     !
6560      Used_mra=INT(FNDefine_used_mra(Requested_mra))
6565      GOSUB Find_mra_index
6570     !
6575     ! ------------------------------------------------------------------------
6580     !
6585     GOSUB Define_time_del  ! Define time delay for that MRA
6590     !
6595     ! ------------------------------------------------------------------------
6600     !
6605     GOSUB Defin_complexwg  ! Define Complex weights for the array
6610     !
```

```
6615      ! ---------------------------------------------------------------
6620      !
6625      !
6630      IF Mode$="DIFF" THEN
6635          GOSUB Set_diff_mode   ! Set DIFFerence mode i.e. window
6640      END IF
6645      !
6650    RETURN  ! End of Array_configure
6655    !
6660    ! --------------------------end-of-sub-routine--------------------------
6665    !
6670    ! ***************************************************************************
6675  Ind_staves_menu:  ! MENU TO CHANGE INDIVIDUAL STAVE COMPLEX SENSITIVITIES *
6680    ! ***************************************************************************
6685    !                                                                           *
6690    ! MODULE NAME: IND_STAVES_MENU                  DATE:  4 NOVEMBER 1985  *
6695    !                                                                           *
6700    ! PROGRAMMER: Sylvain Fleurant                  DATE:  4 NOVEMBER 1985  *
6705    !                                                                           *
6710    ! PURPOSE : To allow the user to change the value of any stave's          *
6715    !           complex weights (amplitude and phase) fo a given MRA.         *
6720    !                                                                           *
6725    ! INPUT: Stave_menu_flag$, Stay_in_menu$,                                 *
6730    ! OUTPUT: Change of the actual complex weights of some hydrophones.      *
6735    ! SIDE EFFECTS: To allow the user to change any stave's complex weights *
6740    !               or sensitivity TEMPORARILY at a given MRA.                *
6745    ! REMARKS: NO CHECK IS PRESENTLY DONE FOR THE HYDROPHONES NOT EVEN        *
6750    !          EXISTING FOR A CONFIGURATION.                                  *
6755    ! ***************************************************************************
6760    !
6765      ! CLEAR THE SCREEN
6770      OUTPUT 2;Clear$;
6775      ! PRINT MESSAGE AND DEMAND IF USER WANT TO USE THIS MENU
6780        Stave_menu_flag$=FNUser_sta_m_choi$  ! Ask if the user needs to use
6785                                             ! this menu.
6790                                             ! User_stave_menu_choice returns
6795                                             ! TRUE for running this menu,
6800                                             ! FALSE for skipping this menu.
6805    ! ---------------------------------------------------------------------
6810      IF Stave_menu_flag$="TRUE " THEN   ! Use the MENU
6815          !
6820        Stay_in_menu$="TRUE "
6825        WHILE (Stay_in_menu$="TRUE ")
6830          GOSUB Output_stavesln !Output on the screen the selection available
6835          GOSUB Enter_stave_nn     ! Allow the user to enter a valid stave #.
6840          !
6845          ! -----------------------------------------------------------------
6850          !
6855          ! INFORM THE USER OF KEY OPTION
6860          !
6865          PRINT "KEY 0 - DISCONNECT A STAVE OR A HYDROPHONE"
6870          PRINT "KEY 1 - STAVE PADDING WITH CAPACITORS"
6875          PRINT "KEY 2 - ALLOW INDIVIDUAL CHANGES OF COMPLEX SENSITIVITY"
6880          PRINT "KEY 9 - EXIT FROM THIS MENU"
6885          PRINT " "
6890          !
6895          ! -----------------------------------------------------------------
6900          !
6905          ! DEFINE MAIN MENU FOR CHOICE OF ACTION TO BE DONE ON THIS STAVE
6910          !
```

```
6915          CONTROL 1,12;0  ! Turn KEY ON
6920          ON KEY 0 LABEL "DISCONNECT" GOTO Disconnect_lbl
6925          ON KEY 1 LABEL "PADDING" GOTO Padding_lbl
6930          ON KEY 2 LABEL "INDIVIDUAL" GOTO Ind_change_lbl
6935          ON KEY 3 GOTO Not_used_sml
6940          ON KEY 4 GOTO Not_used_sml
6945          !
6950          ON KEY 5 GOTO Not_used_sml
6955          ON KEY 6 GOTO Not_used_sml
6960          ON KEY 7 GOTO Not_used_sml
6965          ON KEY 8 GOTO Not_used_sml
6970          ON KEY 9 LABEL "EXIT" GOTO End_stave_menu
6975       ! ------------------------------------------------------------------
6980       !
6985 Stave_m_spinl:   GOTO Stave_m_spinl  ! Wait for softkey interrupt
6990                                      ! - I/O Operator
6995          ! ------------------------------------------------------------------
7000          !
7005 Not_used_sml: !
7010          BEEP 1678,.2  ! This is not a choice
7015          GOTO Stave_m_spinl
7020          !
7025          ! ------------------------------------------------------------------
7030          !
7035 Disconnect_lbl:  !
7040          GOSUB Disconnect_stav
7045          GOTO End_while_menu     ! Any more staves
7050          !
7055          ! ------------------------------------------------------------------
7060          !
7065 Padding_lbl: !
7070          GOSUB Padding_stav
7075          GOTO End_while_menu     ! Any more staves
7080          !
7085          ! ------------------------------------------------------------------
7090          !
7095 Ind_change_lbl: !
7100          GOSUB Ind_stav_change
7105          GOTO End_while_menu     ! Any more staves
7110          !
7115          ! ------------------------------------------------------------------
7120          !
7125 End_while_menu: !
7130          PRINT USING "////"
7135          PRINT "IF YOU DON'T WANT TO CHANGE ANYMORE STAVE"
7140          PRINT "JUST ANSWER THE NEXT QUESTION,"
7145          PRINT "THEN EXIT"
7150          PRINT " "
7155        END WHILE
7160       END IF  ! of Stave_menu_flag$
7165       !
7170 End_stave_menu: !
7175       CONTROL 1,12;1            ! Turn off the softkeys and its display
7180       ! When done clear the screen again
7185       OUTPUT Z;Clear$;
7190      RETURN ! of Ind_staves_menu
7195       !
7200    ! ---------------------------end-of-sub-routine-------------------------
7205    !
7210    ! ***************************************************************************
```

```
7215 Output_stavesln:  ! OUTPUT THE STAVE SELECTION AVAILABLE FOR A GIVEN MRA  *
7220   ! ************************************************************************
7225   !                                                                        *
7230   ! MODULE NAME: OUTPUT_STAVESLN                  DATE:  5 NOVEMBER 1985  *
7235   !                                                                        *
7240   ! PROGRAMMER: Sylvain Fleurant                  DATE:  5 NOVEMBER 1985  *
7245   !                                                                        *
7250   ! PURPOSE : To show to the user the selection of staves available at     *
7255   !           given MRA.                                                   *
7260   !                                                                        *
7265   ! INPUT: Left_1st_array, Left_2nd_array, Right_1st_array,                *
7270   !        Right_2nd_array, Flag_2nd_left$, Flag_2nd_right$, Mra_index,    *
7275   !        Ttl_nn_element, Nn_left_element, Nn_rght_element.               *
7280   ! OUTPUT: list of staves which can be change at a given MRA              *
7285   ! SIDE EFFECTS: NIL                                                      *
7290   ! REMARKS: NIL                                                          *
7295   !                                                                        *
7300   ! ************************************************************************
7305   !
7310      PRINT USING "/"
7315      PRINT "MRA     = ";Used_mra;" degrees"
7320      PRINT "TOTAL NUMBER OF ELEMENT = ";Ttl_nn_element(Mra_index)
7325      PRINT " "
7330      PRINT "THERE ARE ";Nn_left_element(Mra_index);" ELEMENTS IN THE LEFT A
RRAY"
7335      PRINT "                    FROM ";Left_1st_array(Mra_index,1);" TO ";Le
ft_1st_array(Mra_index,2)
7340      IF Flag_2nd_left$(Mra_index)="Y" THEN
7345         IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_index,2)
=0) THEN
7350           ! Only one element is in this array
7355           IF (Left_2nd_array(Mra_index,1)=0) THEN
7360             PRINT "          AND ELEMENT ";Left_2nd_array(Mra_index,2)
7365           ELSE
7370             PRINT "          AND ELEMENT ";Left_2nd_array(Mra_index,1)
7375           END IF
7380         ELSE  ! More than one element
7385            PRINT "               AND FROM ";Left_2nd_array(Mra_index,1);"
 TO ";Left_2nd_array(Mra_index,2)
7390         END IF ! One is zero , i.e., only one element
7395      END IF ! Flag_2nd_array
7400      PRINT " "
7405      PRINT "THERE ARE ";Nn_rght_element(Mra_index);" ELEMENTS IN THE RIGHT
ARRAY"
7410      PRINT "                    FROM ";Right_1st_array(Mra_index,1);" TO ";R
ight_1st_array(Mra_index,2)
7415      IF Flag_2nd_right$(Mra_index)="Y" THEN  ! There is a second array
7420         IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
7425           ! Only one element is in this array
7430           IF (Right_2nd_array(Mra_index,1)=0) THEN
7435             PRINT "          AND ELEMENT ";Right_2nd_array(Mra_index,2)
7440           ELSE
7445             PRINT "          AND ELEMENT ";Right_2nd_array(Mra_index,1)
7450           END IF
7455         ELSE  ! More than one element
7460            PRINT "               AND FROM ";Right_2nd_array(Mra_index,1);
" TO ";Right_2nd_array(Mra_index,2)
7465         END IF ! Only one element in the second array
7470      END IF ! Flag_2nd_right$
```

```
7475      PRINT USING "/"
7480     !NEXT I
7485    RETURN ! Output_stavesln
7490    ! -------------------------end-of-sub-routine-------------------------
7495    !
7500    ! ************************************************************************
7505 Enter_stave_nn:  ! ALLOW USER TO ENTER STAVE NUMBER AS DEFINED PER MRA     *
7510    ! ************************************************************************
7515    !                                                                        *
7520    ! MODULE NAME: ENTER_STAVE_NN                    DATE:  5 NOVEMBER 1985  *
7525    !                                                                        *
7530    ! PROGRAMMER: Sylvain Fleurant                   DATE:  5 NOVEMBER 1985  *
7535    !                                                                        *
7540    ! PURPOSE : To allow the user to enter a stave number to be use by the  *
7545    !           menu to change individual staves' complex sensitivity as     *
7550    !           defined by the current MRA.                                  *
7555    !                                                                        *
7560    ! INPUT: Enter_line$, Ok_input_flag$, Stave_nn_select, Temporary_real,  *
7565    !        Left_1st_array, Left_2nd_array, Right_1st_array,               *
7570    !        Right_2nd_array, Flag_2nd_left$, Flag_2nd_right$, Mra_index.   *
7575    ! OUTPUT: Stave_nn_select                                                *
7580    ! SIDE EFFECTS: NIL                                                     *
7585    ! REMARKS: Will request the user a correct value in the range of the    *
7590    !          stave used at a given MRA.                                   *
7595    ! ************************************************************************
7600    !
7605       CONTROL 1,12;1  ! Disconnect the softkeys, even if it is already done.
7610       Insert_line$=" "        ! Initialize to nothing
7615       !
7620       ! MAKE SURE THAT ALL INPUT DATA ARE OF THE CORRECT TYPE
7625       ! This part will accept any type of input data and process it
7630       !
7635       REPEAT
7640          INPUT "ENTER THE STAVE NUMBER TO CHANGE IN THE RANGE SHOWN ABOVE",
Insert_line$
7645          ! Check if the string is numeric - Pass by value
7650          Ok_input_flag$=FNIs_string_pnum$((Insert_line$))
7655          !
7660          ! ---------------------------------------------------------------
7665          !
7670          IF Ok_input_flag$="TRUE " THEN
7675             ! This is a numeric string
7680             ! Enter the first real number in the string
7685             ! skip all the rest if any characters are left
7690             ENTER Insert_line$;Temporary_real
7695             Stave_nn_select=INT(Temporary_real)
7700          ELSE
7705             ! This is not a POSITIVE number
7710             DISP " YOU MUST ENTER A POSITIVE NUMBER"
7715             WAIT .75   ! Wait 3/4 sec so user can read it
7720          END IF    ! of Ok_input_flag$ is TRUE
7725          !
7730          ! ---------------------------------------------------------------
7735          !
7740          Ok_input_flag$="FALSE"  ! RESET to FALSE to allow for more testing
7745          !
7750          IF ((Stave_nn_select>=Left_1st_array(Mra_index,1)) AND (Stave_nn_s
elect<=Left_1st_array(Mra_index,2))) THEN
7755                ! THIS IS A VALID STAVE NUMBER
7760                Ok_input_flag$="TRUE "
```

```
7765          ELSE
7770            ! ------------------------------------------------------------
7775            IF ((Stave_nn_select)=Right_1st_array(Mra_index,1)) AND (Stave_n
n_select<=Right_1st_array(Mra_index,2))) THEN
7780                 ! THIS IS A VALID STAVE NUMBER
7785                 Ok_input_flag$="TRUE "
7790            ! ------------------------------------------------------------
7795            ELSE
7800              !
7805              !
7810              IF Flag_2nd_left$(Mra_index)="Y" THEN ! Check this array too!
7815                 !
7820                 IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_i
ndex,2)=0) THEN
7825                     ! Only one element is used
7830                     IF (Left_2nd_array(Mra_index,1)=Stave_nn_select) THEN
7835                         Ok_input_flag$="TRUE "
7840                     ELSE
7845                         IF (Left_2nd_array(Mra_index,2)=Stave_nn_select) TH
EN
7850                             Ok_input_flag$="TRUE "
7855                         END IF ! Left_2nd_array for the 2nd - one element
7860                     END IF  ! Left_2nd_array for the 1st - with one element
7865                     !
7870                 ELSE  ! For more than one element in this array
7875                     IF ((Stave_nn_select)=Left_2nd_array(Mra_index,1)) AND
(Stave_nn_select<=Left_2nd_array(Mra_index,2))) THEN
7880                         ! THIS IS A VALID NUMBER
7885                         Ok_input_lag$="TRUE "
7890                     END IF ! Left_2nd_array for more than one element
7895                 END IF ! Left_2nd_array for single element sub-array
7900                 !
7905              END IF ! Flag_2nd_left$ is Y
7910              !
7915              ! ------------------------------------------------------------
7920              !
7925              IF Ok_input_flag$="FALSE" THEN ! More test to do
7930                  !
7935                  IF Flag_2nd_right$(Mra_index)="Y" THEN !Check this array
7940                     !
7945                     ! ------------------------------------------------------
7950                     IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array
(Mra_index,2)=0) THEN
7955                         ! Only one element is used
7960                         IF (Right_2nd_array(Mra_index,1)=Stave_nn_select) T
HEN
7965                             ! This is a valid number
7970                             Ok_input_flag$="TRUE "
7975                         ELSE
7980                             IF (Right_2nd_array(Mra_index,2)=Stave_nn_selec
t) THEN
7985                                 ! This is a valid number
7990                                 Ok_input_flag$="TRUE"
7995                             END IF ! Right_2nd_array(Mra_index,2)
8000                         END IF ! Right_2nd_array(Mra_index,1)
8005                     !
8010                     ! ------------------------------------------------------
8015                     ELSE  ! For more than one element in this array
8020                        IF ((Stave_nn_select)=Right_2nd_array(Mra_index,1))
AND (Stave_nn_select<=Right_2nd_array(Mra_index,2))) THEN
```

```
8025                          ! This is a valid number
8030                          Ok_input_flag$="TRUE "
8035                        END IF ! Stave_nn_select & Right_2nd_array
8040                      END IF ! with Right_2nd_array
8045                    !
8050                    ! ------------------------------------------------------
8055                    !
8060                  END IF ! Flag_2nd_right$ is Y
8065                END IF ! Ok_input_flag$ = "FALSE"
8070                !
8075                ! ------------------------------------------------------
8080                !
8085                IF Ok_input_flag$="FALSE" THEN
8090                     DISP "THIS STAVE NUMBER IS NOT USED FOR THIS MRA"
8095                     WAIT .75  ! so user can read it
8100                END IF ! Ok_input_flag$ = "FALSE" (last case)
8105                !
8110                ! ------------------------------------------------------
8115                !
8120              END IF ! Right_1st_array
8125            END IF ! Left_1st_array
8130          UNTIL Ok_input_flag$="TRUE "     ! i.e. a good frequency
8135          !
8140          ! ------------------------------------------------------------
8145          ! Echo back to the operator
8150          !
8155          DISP "THE STAVE SELECTED IS NUMBER ",Stave_nn_select
8160        !
8165        RETURN ! Enter_stave_nn
8170        !
8175        ! -------------------------end-of-sub-routine-------------------------
8180        !
8185        ! ***************************************************************************
8190 Disconnect_stav:  ! DISCONNECT ONE OR ALL THE HYDROPHONES OF A STAVE      *
8195        ! ***************************************************************************
8200        !                                                                           *
8205        ! MODULE NAME: DISCONNECT_STAVE                    DATE:  5 NOVEMBER 1985 *
8210        !                                                                           *
8215        ! PROGRAMMER: Sylvain Fleurant                     DATE:  6 NOVEMBER 1985 *
8220        !                                                                           *
8225        ! PURPOSE: To disconnect any single hydrophone of a stave or the entire *
8230        !          stave as requested by the operator, for a given pre-selected *
8235        !          valid stave number.                                           *
8240        !                                                                        *
8245        ! INPUT: Hydro_nn, Actual_amp_wg, Stave_nn_select.                       *
8250        ! OUTPUT: Actual_amp_wg                                                  *
8255        ! SIDE EFFECTS: REDEFINE ANY AMPLITUDE WEIGHTS OF THE SUB-ARRAYS        *
8260        !               AS DEFINED FOR A GIVEN MRA                              *
8265        ! REMARKS: NIL                                                          *
8270        ! ***************************************************************************
8275        !
8280          CONTROL 1,12;0  ! Turn KEY ON
8285          OUTPUT Z;Clear$;
8290          Hydro_nn=3      ! All elements are TURN OFF is the default value
8295          !
8300          ! ------------------------------------------------------------------
8305          !
8310          ! INFORM THE USER OF KEY OPTION
8315          !
8320          PRINT USING "///"
```

```
8325      PRINT "KEY 0 - DISCONNECT HYDROPHONE A (TOP)"
8330      PRINT "KEY 1 - DISCONNECT HYDROPHONE B (CENTER)"
8335      PRINT "KEY 2 - DISCONNECT HYDROPHONE C (BOTTOM)"
8340      PRINT "KEY 3 - DISCONNECT ALL HYDROPHONES OF THE STAVE"
8345      PRINT USING "/"
8350      PRINT "KEY 9 - EXIT THIS MENU WITHOUT ANY CHANGES"
8355      DISP "THE STAVE SELECTED IS NUMBER ",Stave_nn_select
8360      !
8365      ! ------------------------------------------------------------------
8370      !
8375      ! DEFINE MAIN MENU FOR CHOICE OF ACTION TO BE DONE ON THIS STAVE
8380      !
8385           ON KEY 0 LABEL "HYDRO A" GOTO Disconnect_1st
8390           ON KEY 1 LABEL "HYDRO B" GOTO Disconnect_2nd
8395           ON KEY 2 LABEL "HYDRO C" GOTO Disconnect_3rd
8400           ON KEY 3 LABEL "ALL HYDRO" GOTO Disconnect_all
8405           ON KEY 4 GOTO Not_used_sm2
8410           !
8415           ON KEY 5 GOTO Not_used_sm2
8420           ON KEY 6 GOTO Not_used_sm2
8425           ON KEY 7 GOTO Not_used_sm2
8430           ON KEY 8 GOTO Not_used_sm2
8435           ON KEY 9 LABEL "EXIT" GOTO End_disconnect
8440      ! ------------------------------------------------------------------
8445      !
8450 Stave_m_spin2:   GOTO Stave_m_spin2  ! Wait for softkey interrupt
8455                                      ! - I/O Operator
8460      ! ------------------------------------------------------------------
8465      !
8470 Not_used_sm2: !
8475        BEEP 1888,.2  ! This is not a choice
8480        GOTO Stave_m_spin2
8485      !
8490      ! ------------------------------------------------------------------
8495      !
8500 Disconnect_1st: !
8505        Hydro_nn=-1
8510        GOTO Disconnect_it
8515      !
8520      ! ------------------------------------------------------------------
8525      !
8530 Disconnect_2nd: !
8535        Hydro_nn=0
8540        GOTO Disconnect_it
8545      !
8550      ! ------------------------------------------------------------------
8555      !
8560 Disconnect_3rd:  !
8565        Hydro_nn=1
8570        GOTO Disconnect_it
8575      !
8580      ! ------------------------------------------------------------------
8585      !
8590 Disconnect_all: !
8595         Hydro_nn=3
8600         GOTO Disconnect_it
8605      !
8610      ! ------------------------------------------------------------------
8615      !
8620 Disconnect_it:  !
```

```
8625      IF Hydro_nn<3 THEN
8630         Actual_amp_wg(Stave_nn_select,Hydro_nn)=0
8635      ELSE  ! All three elements
8640         FOR P=-1 TO 1 STEP 1
8645            Actual_amp_wg(Stave_nn_select,P)=0
8650         NEXT P
8655      END IF ! of Hydro_nn
8660 End_disconnect:  ! Exit label
8665      OUTPUT 2;Clear$;         ! CLEAR THE SCREEN
8670      CONTROL 1,12;1           ! Turn off the softkeys and its display.
8675     RETURN ! Disconnect_stav
8680   ! --------------------------end-of-sub-routine---------------------------
8685   !
8690   ! ************************************************************************
8695 Padding_stav: ! ALLOW USER TO DO PADDING OF ANY STAVE IN THE SUB-ARRAY    *
8700   ! ************************************************************************
8705   !                                                                        *
8710   ! MODULE NAME: PADDING_STAV                      DATE:  6 NOVEMBER 1985 *
8715   !                                                                        *
8720   ! PROGRAMMER: Sylvain Fleurant                   DATE:  6 NOVEMBER 1985 *
8725   !                                                                        *
8730   ! PURPOSE: To allow the user to padd any pre-selected valid stave with  *
8735   !          one or more capacitors.                                       *
8740   !                                                                        *
8745   ! INPUT: Stave_nn_select, Actual_amp_wg, Minimum_nn_db                   *
8750   ! OUTPUT: Actual_amp_wg                                                  *
8755   ! SIDE EFFECTS: Will reset the amplitude weights of the entire stave.   *
8760   ! REMARKS: The value of the amplitude weights will not exceed the -40 dB*
8765   !          or whatever value as defined by minimum_nn_db.                *
8770   !          THIS IMPLEMENTATION SHOULD LATER ON ALLOW VARIATION AS        *
8775   !          FUNCTION OF FREQUENCY.                                        *
8780   ! ************************************************************************
8785   !
8790     ! DEFINE LOCAL VARIABLE
8795     REAL One_padding_amp          ! VALUE OF ONE PADDING
8800     One_padding_amp=-2.8        ! -2.8 dB for any frequency, will be updated
8805     One_padding_amp=10^(One_padding_amp/20)
8810                                 ! Convert it back in linear magnitude.
8815     !
8820     OUTPUT 2;Clear$;              ! Clear the screen
8825     DISP "THE STAVE SELECTED IS NUMBER ",Stave_nn_select
8830     !
8835     ! ---------------------------------------------------------------------
8840     !
8845     ! INFORM THE USER OF KEY OPTION
8850     !
8855     PRINT USING "/"
8860     PRINT "            PADDING MENU"
8865     PRINT " "
8870     PRINT "THIS MENU ALLOWS PADDING OF A STAVE"
8875     PRINT " "
8880     PRINT "KEY 1 - PADDING WITH ONE CAPACITOR"
8885     PRINT "KEY 2 - PADDING WITH TWO CAPACITORS"
8890     PRINT "KEY 3 - PADDING WITH THREE CAPACITORS"
8895     PRINT " "
8900     PRINT "KEY 9 - EXIT THIS MENU WITHOUT CHANGES"
8905     !
8910     ! ---------------------------------------------------------------------
8915     !
8920     ! DEFINE MAIN MENU FOR CHOICE OF ACTION TO BE DONE ON THIS STAVE
```

```
8925     !
8930       CONTROL 1,12;0  ! Turn KEY ON
8935          ON KEY 0 GOTO Not_used_sm3
8940          ON KEY 1 LABEL "1 CAPACITORS" GOTO Padded_1
8945          ON KEY 2 LABEL "2 CAPACITORS" GOTO Padded_2
8950          ON KEY 3 LABEL "3 CAPACITORS" GOTO Padded_3
8955          ON KEY 4 GOTO Not_used_sm3
8960          !
8965          ON KEY 5 GOTO Not_used_sm3
8970          ON KEY 6 GOTO Not_used_sm3
8975          ON KEY 7 GOTO Not_used_sm3
8980          ON KEY 8 GOTO Not_used_sm3
8985          ON KEY 9 LABEL "EXIT" GOTO End_padding_st
8990     ! ------------------------------------------------------------------
8995     !
9000 Stave_m_spin3:   GOTO Stave_m_spin3  ! Wait for softkey interrupt
9005                                      ! - I/O Operator
9010     ! ------------------------------------------------------------------
9015     !
9020 Not_used_sm3: !
9025       BEEP 1678,.2  ! This is not a choice
9030       GOTO Stave_m_spin3
9035     !
9040     ! ------------------------------------------------------------------
9045 Padded_3: ! Three capacitors
9050       FOR P=-1 TO 1 STEP 1
9055          Actual_amp_wg(Stave_nn_select,P)=Actual_amp_wg(Stave_nn_select,P)-O
ne_padding_amp
9060          IF Actual_amp_wg(Stave_nn_select,P)<0 THEN
9065               ! This means it is already to small : 0 --> Minimum_nn_db
9070               Actual_amp_wg(Stave_nn_select,P)=0
9075          END IF
9080       NEXT P
9085     ! The next two will perform the remaining padding (i.e. 2 capacitors)
9090     ! ------------------------------------------------------------------
9095     !
9100 Padded_2: ! Two capacitors
9105       FOR P=-1 TO 1 STEP 1
9110          Actual_amp_wg(Stave_nn_select,P)=Actual_amp_wg(Stave_nn_select,P)-O
ne_padding_amp
9115          IF Actual_amp_wg(Stave_nn_select,P)<0 THEN
9120               ! This means it is already to small : 0 --> Minimum_nn_db
9125               Actual_amp_wg(Stave_nn_select,P)=0
9130          END IF
9135       NEXT P
9140     ! The next one will perform the remaining padding (i.e. one more cap.)
9145     ! ------------------------------------------------------------------
9150     !
9155 Padded_1: ! One capacitor
9160       FOR P=-1 TO 1 STEP 1
9165          Actual_amp_wg(Stave_nn_select,P)=Actual_amp_wg(Stave_nn_select,P)-O
ne_padding_amp
9170          IF Actual_amp_wg(Stave_nn_select,P)<0 THEN
9175               ! This means it is already to small : 0 --> Minimum_nn_db
9180               Actual_amp_wg(Stave_nn_select,P)=0
9185          END IF
9190       NEXT P
9195     !
9200     ! ------------------------------------------------------------------
9205     !
```

```
9210 End_padding_st:   ! Exit label
9215     OUTPUT 2;Clear$;           ! CLEAR THE SCREEN
9220     CONTROL 1,12;1             ! Turn off the softkeys and display
9225     !
9230    RETURN ! from Padding_stav
9235    ! -------------------------end-of-sub-routine--------------------------
9240    !
9245    ! ***************************************************************************
9250 Ind_stav_change: ! CHANGE INDIVIDUAL HYDROPHONE FOR A GIVEN STAVE              *
9255    ! ***************************************************************************
9260    !                                                                           *
9265    ! MODULE NAME: IND_STAVE_CHANGE                  DATE:  6 NOVEMBER 1985   *
9270    !                                                                           *
9275    ! PROGRAMMER: Sylvain Fleurant                   DATE:  6 NOVEMBER 1985   *
9280    !                                                                           *
9285    ! PURPOSE : To change the complex sensitivity or weight (amplitude and      *
9290    !           phase) of any hydrophones of a valid stave within the limit     *
9295    !           of -180 to 180 degrees and 0 to -40 dB (Minimum_nn_db).         *
9300    !                                                                           *
9305    ! INPUT: Stave_nn_select, hydro_nn, Actual_amp_wg, Actual_pha_wg,           *
9310    ! OUTPUT: Actual_amp_wg, Actual_pha_wg                                      *
9315    ! SIDE EFFECTS: Reset value of the sensitivity                              *
9320    ! REMARKS: The changes are permanent during program execution, but no       *
9325    !          changes is done on disk.                                         *
9330    ! ***************************************************************************
9335    !
9340     !
9345     OUTPUT 2;Clear$;              ! Clear the screen
9350     DISP "THE STAVE SELECTED IS NUMBER ",Stave_nn_select
9355     !
9360     ! ---------------------------------------------------------------------
9365     !
9370     ! INFORM THE USER OF KEY OPTION
9375     !
9380     PRINT USING "/"
9385     PRINT "        INDIVIDUAL COMPLEX SENSITIVITY CHANGES"
9390     PRINT " "
9395     PRINT "THIS MENU ALLOWS TO CHANGE THE COMPLEX SENSITIVITY "
9400     PRINT "OF ANY HYDROPHONES OF A PRE-SELECTED STAVE"
9405     PRINT " "
9410     PRINT "KEY 0 - CHANGE COMPLEX SENSITIVITY OF HYDROPHONE A"
9415     PRINT "KEY 1 - CHANGE COMPLEX SENSITIVITY OF HYDROPHONE B"
9420     PRINT "KEY 2 - CHANGE COMPLEX SENSITIVITY OF HYDROPHONE C"
9425     PRINT " "
9430     PRINT "KEY 9 - EXIT THIS MENU WITHOUT CHANGES"
9435     !
9440     ! ---------------------------------------------------------------------
9445     !
9450     ! DEFINE MAIN MENU FOR CHOICE OF ACTION TO BE DONE ON THIS STAVE
9455     !
9460       CONTROL 1,12;0  ! Turn KEY ON
9465          ON KEY 0 LABEL "HYDRO A" GOTO Change_a
9470          ON KEY 1 LABEL "HYDRO B" GOTO Change_b
9475          ON KEY 2 LABEL "HYDRO_C" GOTO Change_c
9480          ON KEY 3 GOTO Not_used_sm4
9485          ON KEY 4 GOTO Not_used_sm4
9490          !
9495          ON KEY 5 GOTO Not_used_sm4
9500          ON KEY 6 GOTO Not_used_sm4
9505          ON KEY 7 GOTO Not_used_sm4
```

```
9510          ON KEY 8 GOTO Not_used_sm4
9515          ON KEY 9 LABEL "EXIT" GOTO End_stav_change
9520     ! ---------------------------------------------------------------------
9525     !
9530 Stave_m_spin4:   GOTO Stave_m_spin4  ! Wait for softkey interrupt
9535                                      ! - I/O Operator
9540     ! ---------------------------------------------------------------------
9545     !
9550 Not_used_sm4: !
9555       BEEP 1789,.2  ! This is not a choice
9560       GOTO Stave_m_spin4
9565     !
9570     ! ---------------------------------------------------------------------
9575     !
9580 Change_a: !
9585       Hydro_nn=-1
9590       GOTO Change_hydro_cw
9595     !
9600     ! ---------------------------------------------------------------------
9605     !
9610 Change_b: !
9615       Hydro_nn=0
9620       GOTO Change_hydro_cw
9625     !
9630     ! ---------------------------------------------------------------------
9635     !
9640 Change_c: !
9645       Hydro_nn=1
9650       GOTO Change_hydro_cw
9655     !
9660     ! ---------------------------------------------------------------------
9665     !
9670 Change_hydro_cw: ! Change individual complex weights
9675     !
9680       ! clear the screen
9685       OUTPUT 2;Clear$;
9690       ! Print current complex sensitivity (weights) before the change
9695       PRINT " ACTUAL COMPLEX SENSITIVITIES BEFORE THE CHANGES"
9700       GOSUB Prt_current_cw
9705       ! CHANGE THE COMPLEX WEIGHTS
9710       GOSUB Cw_hyd_changes
9715       PRINT " MODIFIED COMPLEX WEIGHTS "
9720       GOSUB Prt_current_cw
9725       WAIT 1.5
9730       GOTO Ind_stav_change
9735   !
9740 End_stav_change:  !   Exit label
9745      CONTROL 1,12;1         ! Turn off the softkeys and display
9750      OUTPUT 2;Clear$;       ! Clear the screen
9755    RETURN ! of Ind_stav_change
9760   ! -------------------------end-of-sub-routine---------------------------
9765   !
9770   ! ***************************************************************************
9775 Prt_current_cw:  ! PRINT CURRENT COMPLEX WEIGHT OF AN HYDROPHONE              *
9780   ! ***************************************************************************
9785   !                                                                           *
9790   ! MODULE NAME: PRT_CURRENT_CW                    DATE:  6 NOVEMBER 1985 *
9795   !                                                                           *
9800   ! PROGRAMMER: Sylvain Fleurant                   DATE:  6 NOVEMBER 1985 *
9805   !                                                                           *
```

```
9810   ! PURPOSE : To print the current value of the actual complex sensitivity*
9815   !           of the preselected stave and hydrophone number as call by   *
9820   !           sub-menu to change the individual stave/hydrophone c.weight.*
9825   !                                                                       *
9830   ! INPUT: Stave_nn_select, Hydro_nn, Actual_amp_wg, Actual_pha_wg.       *
9835   ! OUTPUT: Phase and magnitude of complex sensitivity in degrees and     *
9840   !         dB repectively.                                               *
9845   ! SIDE EFFECTS: Data output on the screen.                              *
9850   ! REMARKS: The current value of Hydro_nn and Stave_nn_select are used.  *
9855   ! ************************************************************************
9860   !
9865     !
9870     PRINT " "
9875     PRINT "STAVE # = ";Stave_nn_select;" ,HYDROPHONE # = ";Hydro_nn
9880     Temp_phase=Actual_pha_wg(Stave_nn_select,Hydro_nn)*180/PI
9885     IF Actual_amp_wg(Stave_nn_select,Hydro_nn)<1.E-40 THEN
9890         Temp_mag=Minimum_nn_db
9895     ELSE
9900         Temp_mag=20*LGT(Actual_amp_wg(Stave_nn_select,Hydro_nn))
9905     END IF
9910     IF Temp_amp<Minimum_nn_db THEN
9915         Temp_mag=Minimum_nn_db
9920     END IF
9925     PRINT "MAGNITUDE : ",Temp_mag;" dB"
9930     PRINT "PHASE     : ",Temp_phase;" degrees"
9935     PRINT " "
9940    RETURN ! PRT_CURRENT_CW
9945    !
9950   ! ---------------------------end-of-sub-routine---------------------------
9955   !
9960   ! ************************************************************************
9965 Cw_hyd_changes:  !ENTER AND CHANGE A SPECIFIC HYDROPHONE COMPLEX WEIGHTS   *
9970   ! ************************************************************************
9975   !                                                                       *
9980   ! MODULE NAME: CW_HYDRO_CHANGES                 DATE:  6 NOVEMBER 1985  *
9985   !                                                                       *
9990   ! PROGRAMMER: Sylvain Fleurant                  DATE:  6 NOVEMBER 1985  *
9995   !                                                                       *
10000  ! PURPOSE : To enter a new validated complex weights for a specific     *
10005  !           hydrophone, and then to change it. This is called by the    *
10010  !           sub-menu for individual stave change.                       *
10015  !                                                                       *
10020  ! INPUT: Stave_nn_select, Hydro_nn, Minimum_nn_db, Actual_amp_wg,       *
10025  !        Actual_pha_wg, Temp_phase, Temp_mag.                           *
10030  ! OUTPUT: Actual_amp_wg, Actual_pha_wg.                                 *
10035  ! SIDE EFFECTS: WILL CHANGE THE ACTUAL COMPLEX SENSITIVITIES            *
10040  ! REMARKS: The changes are validated for legality and plausibility.     *
10045  ! ************************************************************************
10050  !
10055    ! INFORM THE USER OF KEY OPTION
10060    !
10065    PRINT " "
10070    PRINT "KEY 0 - CHANGE AMPLITUDE IN dB OF CHOOSEN HYDROPHONE"
10075    PRINT "KEY 1 - CHANGE PHASE IN DEGREES OF CHOOSEN HYDROPHONE"
10080    PRINT "KEY 9 - EXIT THIS MENU WHEN DONE OR IF NO CHANGES ARE REQUIRED"
10085    !
10090    ! ---------------------------------------------------------------------
10095    !
10100    ! DEFINE MAIN MENU FOR CHOICE OF ACTION TO BE DONE ON THIS STAVE
10105    !
```

```
10110       CONTROL 1,12;0  ! Turn KEY ON
10115          ON KEY 0 LABEL "MAGNITUDE" GOTO Change_mag
10120          ON KEY 1 LABEL "  PHASE  " GOTO Change_phase
10125          ON KEY 2 GOTO Not_used_sm5
10130          ON KEY 3 GOTO Not_used_sm5
10135          ON KEY 4 GOTO Not_used_sm5
10140          !
10145          ON KEY 5 GOTO Not_used_sm5
10150          ON KEY 6 GOTO Not_used_sm5
10155          ON KEY 7 GOTO Not_used_sm5
10160          ON KEY 8 GOTO Not_used_sm5
10165          ON KEY 9 LABEL "EXIT" GOTO End_indv_change
10170     ! ---------------------------------------------------------------
10175     !
10180 Stave_m_spin5:  GOTO Stave_m_spin5  ! Wait for softkey interrupt
10185                                     ! - I/O Operator
10190     ! ---------------------------------------------------------------
10195     !
10200 Not_used_sm5:!
10205       BEEP 2300,.2  ! This is not a choice
10210       GOTO Stave_m_spin5
10215     !
10220     ! ---------------------------------------------------------------
10225     !
10230 Change_mag: ! ENTER THE CHANGE - VALIDATE - THEN CHANGE AMPLITUDE WEIGHTS
10235          !
10240          CONTROL 1,12;1           ! TURN OFF SOFTKEYS
10245          !
10250           Insert_line$=" "        ! Initialize to nothing
10255          !
10260          ! MAKE SURE THAT ALL INPUT DATA ARE OF THE CORRECT TYPE
10265          ! This part will accept any type of input data and process it
10270          !
10275          REPEAT
10280              INPUT "ENTER THE MAGNITUDE IN dB (0 TO -40 db) ",Insert_line$
10285              ! Check if the string is numeric - Pass by value
10290              Ok_input_flag$=FNIs_string_num$((Insert_line$))
10295              IF Ok_input_flag$="TRUE " THEN
10300                    ! This is a numeric string
10305                    ! Enter the first real number in the string
10310                    ! skip all the rest if any characters are left
10315                    ENTER Insert_line$;Temporary_real
10320                    Temp_mag=Temporary_real
10325              ELSE
10330                   ! This is not a  number
10335                   DISP " YOU MUST ENTER A NUMBER"
10340                   WAIT .75   ! Wait 3/4 sec so user can read it
10345              END IF    ! of Ok_input_flag$ is TRUE
10350              IF (Temp_mag>0) OR (Temp_mag<Minimum_nn_db) THEN
10355                   Temp_mag=0
10360                   DISP " YOUR VALUE MUST BE BETWEEN 0 AND -40"
10365                   WAIT .75   ! so user can read it
10370                   Ok_input_flag$="FALSE"
10375              END IF  ! of Temp_mag out of range
10380           UNTIL Ok_input_flag$="TRUE "    ! i.e. a good frequency
10385        ! FEEDBACK TO USER
10390          DISP "THE MAGNITUDE ENTERED IS ";Temp_mag;" Degrees"
10395          WAIT .75
10400       !
10405       ! CONVERT IN LINEAR MAGNITUDE
```

```
10410       !
10415       IF Temp_mag=Minimum_nn_db THEN
10420          Temp_mag=0
10425       END IF
10430       Temp_mag=10^(Temp_mag/20)
10435       Actual_amp_wg(Stave_nn_select,Hydro_nn)=Temp_mag
10440       CONTROL 1,12;0    ! TURN ON SOFTKEYS BACK
10445       DISP " "
10450       GOTO Stave_m_spin5
10455     !
10460     ! ------------------------------------------------------------------------
10465     !
10470 Change_phase: ! ENTER THE CHANGE - VALIDATE - THEN CHANGE PHASE WEIGHTS
10475         CONTROL 1,12;1          ! TURN OFF SOFTKEYS
10480         !
10485         Insert_line$=" "        ! Initialize to nothing
10490         !
10495         ! MAKE SURE THAT ALL INPUT DATA ARE OF THE CORRECT TYPE
10500         ! This part will accept any type of input data and process it
10505         !
10510         REPEAT
10515            INPUT "ENTER THE PHASE IN DEGREES (0 TO 360) ",Insert_line$
10520            ! Check if the string is numeric - Pass by value
10525            Ok_input_flag$=FNIs_string_pnum$((Insert_line$))
10530            IF Ok_input_flag$="TRUE " THEN
10535                  ! This is a numeric string
10540                  ! Enter the first real number in the string
10545                  ! skip all the rest if any characters are left
10550                  ENTER Insert_line$;Temporary_real
10555                  Temp_phase=Temporary_real
10560            ELSE
10565                 ! This is not a POSITIVE number
10570                 DISP " YOU MUST ENTER A POSITIVE NUMBER"
10575                 WAIT .75   ! Wait 3/4 sec so user can read it
10580            END IF     ! of Ok_input_flag$ is TRUE
10585            IF (Temp_phase>360) OR (Temp_phase<0) THEN
10590                 Temp_phase=0
10595                 DISP " YOUR VALUE MUST BE BETWEEN 0 AND 360"
10600                 WAIT .75   ! so user can read it
10605                 Ok_input_flag$="FALSE"
10610            END IF  ! of Temp_mag out of range
10615         UNTIL Ok_input_flag$="TRUE "     ! i.e. a good frequency
10620       !
10625       ! FEEDBACK TO USER
10630         DISP "THE PHASE ENTERED IS ";Temp_phase;" Degrees"
10635         WAIT .75
10640       !
10645       ! CONVERT BACK INTO RADIANS
10650       Temp_pha=Temp_pha*(PI/180)
10655       Actual_pha_wg(Stave_nn_select,Hydro_nn)=Temp_phase*PI/180
10660       CONTROL 1,12;0   ! TURN ON SOFTKEYS AGAIN
10665       DISP " "
10670       GOTO Stave_m_spin5
10675     !
10680     ! ------------------------------------------------------------------------
10685     !
10690 End_indv_change: !   Exit label
10695      CONTROL 1,12;1          ! Turn off the softkeys and display
10700      OUTPUT 2;Clear$;        ! Clear the screen
10705   !
```

```
10710   RETURN ! of Cw_hyd_changes
10715   !
10720   ! -------------------------end-of-sub-routine--------------------------
10725   !
10730   ! ****************************************************************************
10735 Beam_pattern:  !   COMPUTE THE BEAM PATTERN OF THE LINE ARRAY                  *
10740   ! ****************************************************************************
10745   !                                                                            *
10750   ! MODULE NAME: BEAM_PATTERN                          DATE: 24 SEPTEMBER 1985 *
10755   !                                                                            *
10760   ! PROGRAMMER: Sylvain Fleurant                       DATE: 25 SEPTEMBER 1985 *
10765   !                                                                            *
10770   ! PURPOSE : Compute the normalized far-field beam pattern and in dB for      *
10775   !           both theoretical and degraded(actual) sub-arrays for a           *
10780   !           given MRA, frequency, speed of sound, and mode of operation.     *
10785   !                                                                            *
10790   ! INPUT   : Psi, Normal_factor, Max_angle, Theory_re_bp, Theory_im_bp,       *
10795   !           Actual_re_bp, Actual_im_bp, Minimum_nn_db, Theory_amp_wg,        *
10800   !           Temporary_real.                                                  *
10805   ! OUTPUT  : Psi, Theory_mag_bp, Theory_phase_bp, Theory_db_bp,               *
10810   !           Actual_mag_bp, Actual_phase_bp, Actual_db_bp, Theory_db_xaxis    *
10815   !           Theory_db_yaxis, Actual_db_xaxis, Actual_db_yaxis                *
10820   ! CALLS   : Polar_form, Rect_form, Find_mra_index, Define_used_mra,          *
10825   !           Theory_dft, Actual_dft, Comp_mag_in_db, Comp_axis_pplot          *
10830   !           Defin_complexwg, Define_time_del                                 *
10835   ! SIDE EFFECTS : NIL                                                         *
10840   ! REMARKS : NIL                                                             *
10845   ! ****************************************************************************
10850   !
10855     !
10860     ! COMPUTE THE REAL AND IMAGINARY PART OF THE BEAM PATTERN
10865     !  THEN NORMALIZE IT AND COMPUTE IT ON A LOGARITHMIC SCALE (dB)
10870     !
10875     RAD
10880     Normal_factor=0
10885     !
10890     ! ---------------------------------------------------------------------
10895     ! DEFINE PSI (HORIZONTAL ANGLE) IN DEGREES
10900     FOR I=-180 TO 180 STEP 1
10905           Psi(I)=I
10910     NEXT I
10915     !
10920     ! CONVERT PSI INTO RADIANS
10925     !
10930     MAT Psi= Psi*(PI/180)
10935     !
10940     ! ---------------------------------------------------------------------
10945     !
10950     ! COMPUTE BEAM PATTERN OF ACTUAL ARRAY
10955     !
10960      T1=TIMEDATE                        ! Starting time of execution of BP
10965      PRINT USING "/"
10970      PRINT "*************************************************************"
10975      PRINT " "
10980      PRINT "THE CALCULATION OF THE THEORETICAL BEAM PATTERN "
10985      PRINT "WILL TAKE ABOUT 2 MIN."
10990      PRINT USING "/"
10995      PRINT "STARTING THEORETICAL DFT"
11000      GOSUB Theory_dft
11005      PRINT "THEORETICAL BEAM PATTERN COMPUTATION DONE"
```

```
11010       PRINT " "
11015       PRINT "THE CALCULATION OF THE ACTUAL ARRAY BEAM PATTERN"
11020       PRINT "WILL ALSO TAKE 2 MIN."
11025       DISP "BEAM PATTERN BEING CALCULATED"
11030       PRINT " "
11035       PRINT "*************************************************************"
11040       PRINT " "
11045       PRINT "STARTING ACTUAL ARRAY DFT"
11050       GOSUB Actual_dft
11055       PRINT "ACTUAL ARRAY BEAM PATTERN COMPUTATION DONE"
11060       PRINT " "
11065       DISP "BEAM PATTERN BEING CALCULATED"
11070       TZ=TIMEDATE                    ! Ending time of execution of BP
11075       PRINT "TIME TO COMPUTE BOTH BP =";TZ-T1;" SECONDS."
11080      !
11085      ! ---------------------------------------------------------------
11090      ! Compute BP in polar form for theoretical and actual array
11095      !
11100      DISP "POLAR FORM BEING COMPUTED"
11105      CALL Polar_form(Theory_re_bp(*),Theory_im_bp(*),Theory_mag_bp(*),Theor
y_phase_bp(*),Max_angle)
11110      CALL Polar_form(Actual_re_bp(*),Actual_im_bp(*),Actual_mag_bp(*),Actua
l_phase_bp(*),Max_angle)
11115      !
11120      !
11125      ! ---------------------------------------------------------------
11130      ! COMPUTE THE NORMALIZATION FACTOR
11135      !
11140      ! Normal_factor=SUM(Theory_amp_wg)  ! Use theoretical data i.e. DC
11145      !
11150      ! USE THE MAXIMUM VALUE OF THE CURVES TO NORMALIZE
11155      !
11160        Temporary_real=Theory_mag_bp(INT(Max_angle)) ! Start with a value
11165        FOR I=-Max_angle TO Max_angle STEP 1
11170            IF (Temporary_real<Theory_mag_bp(I)) THEN
11175                Temporary_real=Theory_mag_bp(I)
11180            END IF
11185            IF (Temporary_real<Actual_mag_bp(I)) THEN
11190                Temporary_real=Actual_mag_bp(I)
11195            END IF
11200        NEXT I
11205        IF Temporary_real>0 THEN
11210           Normal_factor=Temporary_real
11215        END IF
11220      !
11225      ! ---------------------------------------------------------------
11230      ! NORMALIZE THE BEAM PATTERNS
11235      !
11240      IF Normal_factor<>0 THEN
11245           MAT Theory_mag_bp= Theory_mag_bp/(Normal_factor)  ! Normalize BP
11250           MAT Actual_mag_bp= Actual_mag_bp/(Normal_factor)  ! Normalize BP
11255      END IF
11260      !
11265      ! ---------------------------------------------------------------
11270      ! COMPUTE the MAGNITUDE of BP in dB from 0 dB down.
11275      !
11280      DISP "MAGNITUDE IN dB BEING COMPUTED"
11285      CALL Comp_mag_in_db(Theory_db_bp(*),Theory_mag_bp(*),(Minimum_nn_db))
11290      CALL Comp_mag_in_db(Actual_db_bp(*),Actual_mag_bp(*),(Minimum_nn_db))
11295      !
```

```
11300     ! ---------------------------------------------------------------
11305     ! RECONVERT PSI INTO DEGREES
11310     !
11315     MAT Psi= Psi*(180/PI)
11320     !
11325     ! ---------------------------------------------------------------
11330     !
11335     !  Compute BP for the POLAR plot graph
11340     !  for the theoretical and actual array
11345     !
11350     DISP "POLAR REPRESENTATION BEING COMPUTED"
11355     CALL Comp_axis_pplot(Psi(*),Theory_db_xaxis(*),Theory_db_yaxis(*),Theo
ry_db_bp(*),(Minimum_nn_db))
11360     CALL Comp_axis_pplot(Psi(*),Actual_db_xaxis(*),Actual_db_yaxis(*),Actu
al_db_bp(*),(Minimum_nn_db))
11365     !
11370     RAD
11375     DISP " "
11380  !
11385  RETURN
11390  ! -------------------------end-of-sub-routine--------------------------
11395  !
11400  ! ****************************************************************************
11405 Define_time_del: ! DEFINE TIME DELAY FOR A GIVEN MRA                          *
11410  ! ****************************************************************************
11415  !                                                                            *
11420  ! MODULE NAME : DEFINE TIME DELAY              DATE: 29 SEPTEMBRE 1985     *
11425  !                                                                            *
11430  ! PROGRAMMER: SYLVAIN FLEURANT                 DATE: 29 SEPTEMBER 1985     *
11435  !                                                                            *
11440  ! PURPOSE: To define the time delay to be used by  the DFT according        *
11445  !          to a given MRA as implemented by the Beamforming.                 *
11450  !                                                                            *
11455  ! Modification: 1. By S. Fleurant               DATE: 19 OCTOBER 1985      *
11460  !                  To incorporate new values for real time delays for        *
11465  !                  angles 0, +/- 45, and +/- 90 degrees.                     *
11470  !                                                                            *
11475  ! INPUT: Time_delay, Mra_index, MRA Look-up Table.                           *
11480  ! OUTPUT: Time_delay.                                                        *
11485  ! SIDE EFFECTS: NIL                                                         *
11490  ! REMARKS: Time_delay is an array indexed by m.                             *
11495  ! ****************************************************************************
11500    !
11505    ! LOCAL DATA BASE
11510    !
11515    REAL Tdelay              ! Delay count used for the array
11520    REAL Time_delay_cst      ! Constant time delay between elements
11525    Time_delay_cst=6.15E-5           !  61.5 MICROSECONDES
11530    DIM Predefine_td_fl$[5] !Flag to indicate if the pre-defined time delay
11535                             ! database is to be used for phasing
11540                             ! TRUE  = Use the database
11545                             ! FALSE = use the incrementing approach of
11550                             !         constant time delay added.
11555    Predefine_td_fl$="FALSE"
11560    INTEGER Skip_td_count       ! Indicate how many time delay counts have
11565                                ! to be done to move to proper time delay
11570                                ! for the choosen MRA.
11575                                ! i.e. 2 reads twice the data base,
11580                                ! i.e. 5 reads five times the data base...
11585    Skip_td_count=1
```

```
11590    !
11595    ! ------------------------------------------------------------------------
11600    ! DEFINE ACTUAL TIME-DELAY TABLE FOR THE CONFORMAL ARRAY
11605    !
11610    ! Define in secondes, these are the total (cumulative) time delays
11615    ! applied to the staves. They are not inter-staves time delays.
11620    !
11625    ! Values of the TAP number are provided for each staves
11630    ! Since the maximum number of staves used for a given sub-array is 28
11635    ! staves, there will be 28 delays per angle where the ones with zeros
11640    ! are implemented to accelerated the input process.
11645    !
11650    !          ---------------------------------------
11655    !
11660    ! For angle 0 Degree
11665    ! TAP# 21, 28, 33.5, 43, 48.5, 55, 62, 68.5
11670    DATA 0.0,-430.5,-768.75,-1353.0,-1691.25,-2091.0,-2521.5,-2921.25
11675    ! TAP# 75,81.5, 87, 92.5, 97.5, 101.5
11680    DATA -3321.0,-3720.75,-4059.0,-4397.25, -4704.75, -4950.75
11685    ! TAP# 101.5, 97.5, 92.5, 87, 81.5, 75, 68.5
11690    DATA -4950.75,-4704.75,-4397.25,-4059.0,-3720.75,-3321.,-2921.25
11695    ! TAP# 62, 55, 48.5, 43, 33.5, 28, 21
11700    DATA -2521.5,-2091.0,-1691.25,-1353.0,-768.75,-430.5, 0.0
11705    !
11710    !  THEORETICAL TIME DELAYS FOR A SPEED OF 1402.08 M/S (4600 FT/SEC)
11715    ! DATA 0.0, 422.59, 772.75, 1355.29, 1688.18, 2103.71, 2515.85, 2920.06
11720    ! DATA 3318.04, 3730.75, 4073.82, 4410.10, 4735.05, 4966.03
11725    ! DATA 4966.03, 4735.05, 4410.1, 4073.82, 3730.75, 3318.04, 2920.06
11730    ! DATA 2515.85, 2103.71, 1688.18, 1355.29, 772.75, 422.59, 0.0
11735    !
11740    !          ---------------------------------------
11745    !
11750    ! For +/- 45 Degrees (At a speed of 1402.08 m/sec.) (Stave 37 is 0)
11755    ! TAP# 66, 72.5, 77.5, 79, 79, 78, 76.5, 74
11760    DATA -738.0,-1137.75,-1445.25,-1537.5,-1537.5,-1476.0,-1383.75,-1230.0
11765    ! TAP# 71.5, 68.5, 65.5, 62, 59
11770    DATA -1076.25,-891.75, -707.25, -492.0, -307.5
11775    ! TAP# 54, 50.5, 46.5, 43, 38.5, 34.5
11780    DATA 0.0, 215.25, 461.25, 676.5, 953.25, 1199.25
11785    ! TAP# 30, 25.5, 21.5, 17, 13, 8, 4, 0 , 0
11790    DATA 1476.0, 1752.75, 1998.75, 2275.5, 2521.5, 2829.0, 3075.0, 0 ,0
11795    !
11800    !          ---------------------------------------
11805    !
11810    ! For +/- 90 Degrees (At a speed of 1402.08 m/sec.)
11815    ! TAP# 15, 21, 26, 30, 33.5, 36.5, 39.5, 42
11820    DATA  2275.5, 1906.5, 1599.0, 1353.0, 1137.75, 953.25, 768.75, 615.0
11825    ! TAP# 44, 46, 47, 49.5, 51
11830    DATA 492.0,369.0,307.5,153.75,61.5
11835    ! TAP# 52, 53, 54.4, 55.5, 56.5, 57.5, 58, 59, 60
11840    DATA 0,-61.5,-153.75,-215.25,-276.75,-338.25,-369.0,-430.5,-492.0
11845    ! TAP 61, 61, 62, 62.5, 0, 0
11850    DATA -553.5,-553.5, -615.0, -645.75, 0 ,0
11855    !
11860    !          ---------------------------------------
11865    IF (Used_mra)-5) AND (Used_mra<5.0) THEN
11870       Predefine_td_fl$="TRUE "
11875       Skip_td_count=1      ! First data group of time delays
11880    ELSE
11885       IF (ABS(Used_mra)=45.0) THEN
```

```
11890       Predefine_td_fl$="TRUE "
11895       Skip_td_count=2     ! Second data group of time delays
11900       ELSE
11905           IF (ABS(Used_mra)=90.0) THEN
11910               Predefine_td_fl$="TRUE "
11915               Skip_td_count=3     ! Third data group of time delays
11920           !ELSE
11925           !   IF (ABS(Used_mra)=135.0) THEN
11930           !       Predefine_td_fl$="TRUE "
11935           !       Skip_td_count=4   ! Fourth data group of time delays
11940           !   END IF ! of +/- 135 degrees
11945           END IF ! of +/- 90 degrees
11950       END IF ! of +/- 45 degrees
11955     END IF ! of 0 degrees
11960     IF (Predefine_td_fl$="TRUE ") THEN
11965       ! ENTER THE ACTUAL TIME DELAY SINCE THEY ARE DEFINED
11970       FOR J=1 TO Skip_td_count STEP 1
11975           READ Time_del_buffer(*)
11980       NEXT J
11985       !
11990       ! ------------------------------------------------------------------
11995       ! Adjust the time delays for negative MRA
12000       !
12005       IF Used_mra<0 THEN
12010           MAT Temp_td_buffer= Time_del_buffer   ! COPY TIME DELAY
12015           ! Skip the unused zero
12020           K=(Nn_left_element(Mra_index))
12025           ! MAKE AN IMAGE COPY FOR THE NEGATIVE MRA
12030           ! DO THE LEFT ARRAY - (A simple inversion)
12035           FOR J=1 TO K STEP 1
12040               Time_del_buffer(J)=Temp_td_buffer(K-J+1)
12045           NEXT J
12050           ! DO THE RIGHT ARRAY
12055           M=Ttl_nn_element(Mra_index)
12060           FOR J=K+1 TO M STEP 1
12065              Time_del_buffer(J)=Temp_td_buffer(M-J+K+1)
12070           NEXT J
12075           K=0
12080           M=0   ! reset
12085       END IF  ! Used_mra < 0
12090       !
12095       ! ------------------------------------------------------------------
12100       ! For debugging purpose
12105       !
12110       ! FOR J=1 TO 28
12115       !    PRINT "J = ";J;"  TIME DELAY = ";Time_del_buffer(J)
12120       ! NEXT J
12125       ! PAUSE
12130       !
12135       ! ------------------------------------------------------------------
12140       !
12145       ! The correct time delay buffer can now be used for the choosen MRA
12150       I=1 ! Initialize index of Time_del_buffer
12155       ! ------------------------------------------------------------------
12160       ! FIRST LEFT SUB-ARRAY   -  NEGATIVE TIME DELAY
12165       !
12170       FOR M=Left_1st_array(Mra_index,1) TO Left_1st_array(Mra_index,2) STEP
 1
12175               Time_delay(M)=Time_del_buffer(I)
12180               I=I+1
```

```
12185       NEXT M
12190       !
12195       ! ---------------------------------------------------------------
12200       ! SECOND LEFT SUB-ARRAY - DEACTIVATED FEATURES SINC NONE EXIST HERE
12205       !
12210       IF Left_activate$="TRUE " THEN
12215       IF Flag_2nd_left$(Mra_index)="Y" THEN   ! There is a sub-array
12220          IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_index,2)
=0) THEN
12225          ! Only one element is used
12230             IF (Left_2nd_array(Mra_index,1)=0) THEN
12235                M=Left_2nd_array(Mra_index,2)
12240             ELSE
12245                M=Left_2nd_array(Mra_index,1)
12250             END IF
12255             Time_delay(M)=Time_del_buffer(I)
12260             I=I+1
12265             !
12270          ELSE  ! For more than one element in this array
12275              FOR M=Left_2nd_array(Mra_index,1) TO Left_2nd_array(Mra_index,
2) STEP 1
12280                 Time_delay(M)=Time_del_buffer(I)
12285                 I=I+1
12290              NEXT M
12295              !
12300          END IF ! with Left_2nd_array
12305       END IF  ! for Flag_2nd_left$
12310       END IF  ! for Left_activate$
12315       !
12320       ! ---------------------------------------------------------------
12325       ! FIRST RIGHT SUB-ARRAY    -    POSITIVE TIME DELAY
12330       !
12335       FOR M=Right_1st_array(Mra_index,1) TO Right_1st_array(Mra_index,2) ST
EP 1
12340                 Time_delay(M)=Time_del_buffer(I)
12345                 I=I+1
12350       NEXT M
12355       !
12360       ! ---------------------------------------------------------------
12365       ! SECOND RIGHT SUB-ARRAY
12370       !
12375       IF Flag_2nd_right$(Mra_index)="Y" THEN   ! There is a sub-array
12380          IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
12385          ! Only one element is used
12390             IF (Right_2nd_array(Mra_index,1)=0) THEN
12395                M=Right_2nd_array(Mra_index,2)
12400             ELSE
12405                M=Right_2nd_array(Mra_index,1)
12410             END IF
12415             Time_delay(M)=Time_del_buffer(I)
12420             I=I+1
12425             !
12430          ELSE  ! For more than one element in this array
12435              FOR M=Right_2nd_array(Mra_index,1) TO Right_2nd_array(Mra_inde
x,2) STEP 1
12440                 Time_delay(M)=Time_del_buffer(I)
12445                 I=I+1
12450              NEXT M
12455              !
```

```
12460         END IF ! with Right_2nd_array
12465       END IF  ! for Flag_2nd_right$
12470       I=1  !Reset the index of the time delay buffer
12475       MAT Time_delay= Time_delay*(1.E-6)
12480       !
12485       !
12490     ELSE !     ****************************************
12495      ! TEMPORARILY USED THE CUMULATIVE 61.5 MICRO-SECONDS
12500       !
12505       ! ---------------------------------------------------------------
12510       ! FIRST LEFT SUB-ARRAY   -  NEGATIVE TIME DELAY
12515       !
12520       FOR M=Left_1st_array(Mra_index,2) TO Left_1st_array(Mra_index,1) STEP
 -1
12525               Tdelay=Tdelay+Time_delay_cst
12530               Time_delay(M)=-Tdelay
12535       NEXT M
12540       !
12545       ! ---------------------------------------------------------------
12550       ! SECOND LEFT SUB-ARRAY - DEACTIVATED FEATURES SINC NONE EXIST HERE
12555       !
12560       IF Left_activate$="TRUE " THEN
12565       IF Flag_2nd_left$(Mra_index)="Y" THEN   ! There is a sub-array
12570          IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_index,2)
=0) THEN
12575          ! Only one element is used
12580             IF (Left_2nd_array(Mra_index,1)=0) THEN
12585                  M=Left_2nd_array(Mra_index,2)
12590             ELSE
12595                  M=Left_2nd_array(Mra_index,1)
12600             END IF
12605             Tdelay=Tdelay+Time_delay_cst
12610             Time_delay(M)=-Tdelay
12615             !
12620          ELSE  ! For more than one element in this array
12625              FOR M=Left_2nd_array(Mra_index,2) TO Left_2nd_array(Mra_index,
1) STEP -1
12630                   Tdelay=Tdelay+Time_delay_cst
12635                   Time_delay(M)=-Tdelay
12640              NEXT M
12645              !
12650          END IF ! with Left_2nd_array
12655       END IF  ! for Flag_2nd_left$
12660       END IF  ! for Left_activate$
12665       !
12670       ! ---------------------------------------------------------------
12675       !
12680       Tdelay=0.          ! RESET delay for other half array
12685       !
12690       ! ---------------------------------------------------------------
12695       ! FIRST RIGHT SUB-ARRAY    -    POSITIVE TIME DELAY
12700       !
12705       FOR M=Right_1st_array(Mra_index,1) TO Right_1st_array(Mra_index,2) ST
EP 1
12710               Tdelay=Tdelay+Time_delay_cst
12715               Time_delay(M)=Tdelay
12720       NEXT M
12725       !
12730       ! ---------------------------------------------------------------
12735       ! SECOND RIGHT SUB-ARRAY
```

```
12740      !
12745      IF Flag_2nd_right$(Mra_index)="Y" THEN   ! There is a sub-array
12750         IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
12755         ! Only one element is used
12760            IF (Right_2nd_array(Mra_index,1)=0) THEN
12765                 M=Right_2nd_array(Mra_index,2)
12770            ELSE
12775                 M=Right_2nd_array(Mra_index,1)
12780            END IF
12785            Tdelay=Tdelay+Time_delay_cst
12790            Time_delay(M)=Tdelay
12795            !
12800         ELSE  ! For more than one element in this array
12805             FOR M=Right_2nd_array(Mra_index,1) TO Right_2nd_array(Mra_inde
x,2) STEP 1
12810                  Tdelay=Tdelay+Time_delay_cst
12815                  Time_delay(M)=Tdelay
12820             NEXT M
12825             !
12830         END IF ! with Right_2nd_array
12835      END IF  ! for Flag_2nd_right$
12840      !
12845      !
12850    END IF ! Used_mra = Angle with defined actual time delay
12855      !
12860      ! For printing the time delays applied to all elements
12865      !
12870      ! PRINT USING "//"
12875      ! FOR I=1 TO 52 STEP 1
12880           ! PRINT I,Time_delay(I)
12885      ! NEXT I
12890      ! BEEP 888,.3
12895      ! PAUSE     ! Wait until the user had read it
12900    RETURN ! Define_time_delay
12905  ! -------------------------end-of-sub-routine-------------------------
12910  !
12915  ! ****************************************************************************
12920 Defin_complexwg: ! DEFINE THE COMPLEX WEIGHTS FOR THE ACTIVATED ELEMENT  *
12925  ! ****************************************************************************
12930  !                                                                          *
12935  ! MODULE NAME : DEFINE COMPLEX WEIGHTS        DATE: 29 SEPTEMBRE 1985   *
12940  !                                                                          *
12945  ! PROGRAMMER: SYLVAIN FLEURANT                DATE: 29 SEPTEMBER 1985   *
12950  !                                                                          *
12955  ! PURPOSE: To define the complex weights  used by  the DFT according     *
12960  !          to a given MRA for both theoretical and actual arrays.         *
12965  !                                                                          *
12970  ! INPUT: Mra_index, MRA Look-up Table, Actual_pha_wg, Actual_amp_wg,      *
12975  !        Theory_pha_wg, Theory_amp_wg                                      *
12980  ! OUTPUT: Actual_pha_wg, Actual_amp_wg, Theory_pha_wg, Theory_amp_wg      *
12985  ! SIDE EFFECTS: NIL                                                        *
12990  ! REMARKS: 1. The complex weights are indexed by m & p.                   *
12995  !                                                                          *
13000  !          2. The amplitude weights are normalize to 1.0, and the phase   *
13005  !          weights are in radians from 0 to 2PI. Not using this will      *
13010  !          result in incorrect beam pattern.                              *
13015  !                                                                          *
13020  !          3. The complex weights can be read as sensitivitities of       *
13025  !          individual elements from a data file for a given               *
```

```
13030  !          frequency. The amplitude should be in dB (0 dB down) and the *
13035  !          phase in degrees.  They must first be converted.              *
13040  ! ************************************************************************
13045  !
13050       MAT Amp_wg_buffer= (1.0)  ! Actual same as theoretical unless real
13055                                 ! values are required
13060       MAT Pha_wg_buffer= (0.)
13065       Read_cs_flag$="N"
13070       INPUT "DO YOU WANT THE ACTUAL SENSITIVITIES FROM DISK INSTEAD OF DEFA
ULT (Y or N)?",Read_cs_flag$
13075       IF (Read_cs_flag$="y") OR (Read_cs_flag$="Y") THEN
13080          CALL Read_actual_cs((Freq),Amp_wg_buffer(*),Pha_wg_buffer(*),(Mini
mum_nn_db),(Max_nn_staves)) !
13085       END IF
13090       !
13095       ! FIST LEFT SUB-ARRAY
13100       !
13105       FOR M=Left_1st_array(Mra_index,1) TO Left_1st_array(Mra_index,2)
13110          FOR P=-1 TO 1 STEP 1
13115                Theory_amp_wg(M,P)=1.0
13120                Theory_pha_wg(M,P)=0.
13125                !
13130                ! TEMPORARY FOR ACTUAL
13135                Actual_amp_wg(M,P)=Amp_wg_buffer(M)
13140                Actual_pha_wg(M,P)=Pha_wg_buffer(M)
13145          NEXT P
13150       NEXT M
13155       !
13160       ! ---------------------------------------------------------------
13165       ! FIRST RIGHT SUB-ARRAY
13170       FOR M=Right_1st_array(Mra_index,1) TO Right_1st_array(Mra_index,2)
13175          !
13180          FOR P=-1 TO 1 STEP 1
13185                Theory_amp_wg(M,P)=1.0
13190                Theory_pha_wg(M,P)=0.
13195                ! TEMPORARY FOR ACTUAL
13200                !
13205                Actual_amp_wg(M,P)=Amp_wg_buffer(M)
13210                Actual_pha_wg(M,P)=Pha_wg_buffer(M)
13215          NEXT P
13220          !
13225       NEXT M
13230       !
13235       ! ---------------------------------------------------------------
13240       ! SECOND LEFT SUB-ARRAY - DEACTIVATED FEATURES SINC NONE EXIST HERE
13245       !
13250       IF Left_activate$="TRUE " THEN
13255       IF Flag_2nd_left$(Mra_index)="Y" THEN   ! There is a sub-array
13260          IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_index,2)
=0) THEN
13265          ! Only one element is used
13270             IF (Left_2nd_array(Mra_index,1)=0) THEN
13275                M=Left_2nd_array(Mra_index,2)
13280             ELSE
13285                M=Left_2nd_array(Mra_index,1)
13290             END IF
13295             FOR P=-1 TO 1 STEP 1
13300                Theory_amp_wg(M,P)=1.0
13305                Theory_pha_wg(M,P)=0.
13310                ! TEMPORARY FOR ACTUAL
```

```
13315                !
13320                Actual_amp_wg(M,P)=Amp_wg_buffer(M)
13325                Actual_pha_wg(M,P)=Pha_wg_buffer(M)
13330             NEXT P
13335             !
13340          ELSE  ! For more than one element in this array
13345             FOR M=Left_2nd_array(Mra_index,1) TO Left_2nd_array(Mra_index,
2)
13350                FOR P=-1 TO 1 STEP 1
13355                     Theory_amp_wg(M,P)=1.0
13360                     Theory_pha_wg(M,P)=0.
13365                     ! TEMPORARY FOR ACTUAL
13370                     !
13375                     Actual_amp_wg(M,P)=Amp_wg_buffer(M)
13380                     Actual_pha_wg(M,P)=Pha_wg_buffer(M)
13385                NEXT P
13390             NEXT M
13395             !
13400          END IF ! with Left_2nd_array
13405       END IF  ! for Flag_2nd_left$
13410       END IF  ! for Left_activate$
13415       !
13420       ! ---------------------------------------------------------------
13425       ! SECOND RIGHT SUB-ARRAY
13430       IF Flag_2nd_right$(Mra_index)="Y" THEN   ! There is a sub-array
13435          IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
13440          ! Only one element is used
13445             IF (Right_2nd_array(Mra_index,1)=0) THEN
13450                M=Right_2nd_array(Mra_index,2)
13455             ELSE
13460                M=Right_2nd_array(Mra_index,1)
13465             END IF
13470             FOR P=-1 TO 1 STEP 1
13475                      Theory_amp_wg(M,P)=1.0
13480                      Theory_pha_wg(M,P)=0.
13485                      ! TEMPORARY FOR ACTUAL
13490                      !
13495                      Actual_amp_wg(M,P)=Amp_wg_buffer(M)
13500                      Actual_pha_wg(M,P)=Pha_wg_buffer(M)
13505             NEXT P
13510             !
13515          ELSE  ! For more than one element in this array
13520             FOR M=Right_2nd_array(Mra_index,1) TO Right_2nd_array(Mra_inde
x,2)
13525                FOR P=-1 TO 1 STEP 1
13530                     Theory_amp_wg(M,P)=1.0
13535                     Theory_pha_wg(M,P)=0.
13540                     ! TEMPORARY FOR ACTUAL
13545                     !
13550                     Actual_amp_wg(M,P)=Amp_wg_buffer(M)
13555                     Actual_pha_wg(M,P)=Pha_wg_buffer(M)
13560                NEXT P
13565             NEXT M
13570             !
13575          END IF ! with Right_2nd_array
13580       END IF  ! for Flag_2nd_right$
13585       !
13590       !
13595       ! ---------------------------------------------------------------
```

```
13600       ! TURN THE HYDROPHONES WHICH ARE ABSENT FOR A GIVEN CONFIGURATION
13605       !
13610       Theory_amp_wg(21,0)=0.      ! Similar to a disconnect element
13615       Actual_amp_wg(21,0)=0.      !   STAVE # 21, element B
13620       !
13625       Theory_amp_wg(25,-1)=0.
13630       Actual_amp_wg(25,-1)=0.     !   STAVE # 25,
13635       Theory_amp_wg(25,1)=0.      !   elements A & C
13640       Actual_amp_wg(25,1)=0.
13645       !
13650       Theory_amp_wg(26,-1)=0.
13655       Actual_amp_wg(26,-1)=0.     !   STAVE # 26,
13660       Theory_amp_wg(26,1)=0.      !   elements A & C
13665       Actual_amp_wg(26,1)=0.
13670       !
13675       Theory_amp_wg(27,-1)=0.
13680       Actual_amp_wg(27,-1)=0.     !   STAVE # 27,
13685       Theory_amp_wg(27,1)=0.      !   element A & C
13690       Actual_amp_wg(27,1)=0.
13695       !
13700       Theory_amp_wg(28,-1)=0.
13705       Actual_amp_wg(28,-1)=0.     !   STAVE # 28,
13710       Theory_amp_wg(28,1)=0.      !   element A & C
13715       Actual_amp_wg(28,1)=0.
13720       !
13725     RETURN ! Defin_complexwg
13730   ! -------------------------end-of-sub-routine--------------------------
13735   !
13740   ! ****************************************************************************
13745 Set_diff_mode: ! SET DIFFERENCE MODE OF OPERATION                              *
13750   ! ****************************************************************************
13755   !                                                                            *
13760   ! MODULE NAME : SET DIFFERENCE MODE           DATE: 2 OCTOBER 1985         *
13765   !                                                                            *
13770   ! PROGRAMMER: SYLVAIN FLEURANT                DATE: 2 OCTOBER 1985         *
13775   !                                                                            *
13780   ! PURPOSE: To add an additional 180 degrees of phase to every elements       *
13785   !          define in the right sub-arrays (1st and 2nd too) to generate      *
13790   !          a difference window on the complex weights.                       *
13795   !                                                                            *
13800   ! INPUT: Mra_index, MRA Look-up Table, Actual_pha_wg, Theory_pha_wg.         *
13805   ! OUTPUT: Actual_pha_wg, Theory_pha_wg.                                      *
13810   ! SIDE EFFECTS: NIL                                                         *
13815   ! REMARKS: 1. The complex weights are indexed by m & p.                     *
13820   !                                                                            *
13825   !          2. The amplitude weights are normalize to 1.0, and the phase      *
13830   !          weights are in radians from 0 to 2PI. Not using this will         *
13835   !          result in incorrect beam pattern.                                 *
13840   !                                                                            *
13845   !          3. The additional phase changes is pi radians (not 180            *
13850   !          degrees per say).                                                 *
13855   ! ****************************************************************************
13860       !
13865       ! FIST RIGHT SUB-ARRAY
13870       !
13875       FOR M=Right_1st_array(Mra_index,1) TO Right_1st_array(Mra_index,2)
13880          FOR P=-1 TO 1 STEP 1
13885               Theory_pha_wg(M,P)=Theory_pha_wg(M,P)+PI
13890               Actual_pha_wg(M,P)=Actual_pha_wg(M,P)+PI
13895          NEXT P
```

```
13900       NEXT M
13905       !
13910       ! ------------------------------------------------------------------
13915       ! SECOND RIGHT SUB-ARRAY
13920       IF Flag_2nd_right$(Mra_index)="Y" THEN   ! There is a sub-array
13925          IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
13930          ! Only one element is used
13935             IF (Right_2nd_array(Mra_index,1)=0) THEN
13940                M=Right_2nd_array(Mra_index,2)
13945             ELSE
13950                M=Right_2nd_array(Mra_index,1)
13955             END IF
13960             FOR P=-1 TO 1 STEP 1
13965                   Theory_pha_wg(M,P)=Theory_pha_wg(M,P)+PI
13970                   Actual_pha_wg(M,P)=Actual_pha_wg(M,P)+PI
13975              NEXT P
13980             !
13985          ELSE  ! For more than one element in this array
13990              FOR M=Right_2nd_array(Mra_index,1) TO Right_2nd_array(Mra_inde
x,2)
13995                 FOR P=-1 TO 1 STEP 1
14000                    Theory_pha_wg(M,P)=Theory_pha_wg(M,P)+PI
14005                    Actual_pha_wg(M,P)=Actual_pha_wg(M,P)+PI
14010                 NEXT P
14015              NEXT M
14020              !
14025          END IF ! with Right_2nd_array
14030       END IF  ! for Flag_2nd_right$
14035       !
14040    !
14045    RETURN ! Set_diff_mode
14050    ! ------------------------end-of-sub-routine----------------------------
14055    !
14060    ! ***************************************************************************
14065 Theory_dft: ! COMPUTE THE DFT OF THE THEORETICAL ARRAY                        *
14070    ! ***************************************************************************
14075    !                                                                          *
14080    ! MODULE NAME : THEORY_DFT                    DATE: 29 SEPTEMBRE 1985    *
14085    !                                                                          *
14090    ! PROGRAMMER: SYLVAIN FLEURANT                DATE: 29 SEPTEMBER 1985    *
14095    !                                                                          *
14100    ! PURPOSE: To compute the unnormalized real and imaginary parts of a      *
14105    !          beam pattern for a specific MRA, Sensitivity, wavelength, &     *
14110    !          frequency for the theoretical array (with correct               *
14115    !          sensitivity).                                                   *
14120    !                                                                          *
14125    ! INPUT: Actual_amp_wg, Actual_pha_wg, X_psn, Y_psn, Time_delay, Psi,     *
14130    !        Mra_index, freq, & lambda.                                        *
14135    ! OUTPUT: Theory_re_bp, Theory_im_bp                                      *
14140    ! SIDE EFFECTS: NIL                                                       *
14145    ! REMARKS: 1. The MRA look_up table is considered to be global i.e. it    *
14150    !         can access by this sub-routine.                                  *
14155    !                                                                          *
14160    !         2. Up to four sections can be executed, allowing up to four     *
14165    !         sub-arrays to be considered in the calculation of the beam       *
14170    !         pattern.                                                         *
14175    !                                                                          *
14180    !         3. In the present implementation, there is no left second        *
14185    !         sub-array. It is implemented but disconnected as comments.       *
```

```
14190  !         Future software development may require it uses.              *
14195  !                                                                        *
14200  !         4. Some redundancies do exist in order to accelerate the     *
14205  !         computation speed, instead of using more sub-routines calls  *
14210  !         and context switching.                                       *
14215  ! ************************************************************************
14220   !
14225   ! LOCAL DATA BASE DEFINITION
14230   !
14235    REAL Phase_m                  ! Phase changes in m.
14240    REAL Phase_mp                 ! Phase changes in m (Phase_m) and p.
14245   !
14250   ! ---------------------------------------------------------------------
14255   !
14260   ! COMPUTE THE DFT FOR ALL PSI (Horizontal angles)
14265   !
14270    RAD
14275    FOR I=-180 TO 180 STEP 1
14280      !
14285      ! ---------------------------------------------------------------
14290      ! FIRST LEFT SUB_ARRAY
14295      !
14300      FOR M=Left_1st_array(Mra_index,1) TO Left_1st_array(Mra_index,2)
14305         Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(I))
*Y_psn(M))/Lambda)
14310         FOR P=-1 TO 1 STEP 1
14315            Phase_mp=Phase_m+Theory_pha_wg(M,P)
14320            IF Theory_amp_wg(M,P)<>0 THEN
14325               Theory_re_bp(I)=Theory_re_bp(I)+Theory_amp_wg(M,P)*COS(Phase
_mp)
14330               Theory_im_bp(I)=Theory_im_bp(I)+Theory_amp_wg(M,P)*SIN(Phase
_mp)
14335            END IF
14340         NEXT P
14345      NEXT M
14350      !
14355      ! ---------------------------------------------------------------
14360      ! FIRST RIGHT SUB-ARRAY
14365      FOR M=Right_1st_array(Mra_index,1) TO Right_1st_array(Mra_index,2)
14370         !
14375         Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(I))
*Y_psn(M))/Lambda)
14380         FOR P=-1 TO 1 STEP 1
14385            Phase_mp=Phase_m+Theory_pha_wg(M,P)
14390            IF Theory_amp_wg(M,P)<>0 THEN
14395               Theory_re_bp(I)=Theory_re_bp(I)+Theory_amp_wg(M,P)*COS(Phase
_mp)
14400               Theory_im_bp(I)=Theory_im_bp(I)+Theory_amp_wg(M,P)*SIN(Phase
_mp)
14405            END IF
14410         NEXT P
14415         !
14420      NEXT M
14425      !
14430      ! ---------------------------------------------------------------
14435      ! SECOND LEFT SUB-ARRAY - DEACTIVATED FEATURES SINC NONE EXIST HERE
14440      !
14445      IF Left_activate$="TRUE " THEN
14450      IF Flag_2nd_left$(Mra_index)="Y" THEN   ! There is a sub-array
14455         IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_index,2)
```

```
=0) THEN
14460         ! Only one element is used
14465            IF (Left_2nd_array(Mra_index,1)=0) THEN
14470                 M=Left_2nd_array(Mra_index,2)
14475            ELSE
14480                 M=Left_2nd_array(Mra_index,1)
14485            END IF
14490            Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(
I))*Y_psn(M))/Lambda)
14495            FOR P=-1 TO 1 STEP 1
14500                Phase_mp=Phase_m+Theory_pha_wg(M,P)
14505                IF Theory_amp_wg(M,P)<>0 THEN
14510                  Theory_re_bp(I)=Theory_re_bp(I)+Theory_amp_wg(M,P)*COS(Ph
ase_mp)
14515                  Theory_im_bp(I)=Theory_im_bp(I)+Theory_amp_wg(M,P)*SIN(Ph
ase_mp)
14520                END IF
14525             NEXT P
14530            !
14535         ELSE  ! For more than one element in this array
14540            FOR M=Left_2nd_array(Mra_index,1) TO Left_2nd_array(Mra_index,
2)
14545               Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(
Psi(I))*Y_psn(M))/Lambda)
14550               FOR P=-1 TO 1 STEP 1
14555                  Phase_mp=Phase_m+Theory_pha_wg(M,P)
14560                  IF Theory_amp_wg(M,P)<>0 THEN
14565                    Theory_re_bp(I)=Theory_re_bp(I)+Theory_amp_wg(M,P)*CO
S(Phase_mp)
14570                    Theory_im_bp(I)=Theory_im_bp(I)+Theory_amp_wg(M,P)*SI
N(Phase_mp)
14575                  END IF
14580               NEXT P
14585            NEXT M
14590            !
14595         END IF ! with Left_2nd_array
14600      END IF  ! for Flag_2nd_left$
14605      END IF  ! for Left_activate$
14610      !
14615      ! ------------------------------------------------------------------
14620      ! SECOND RIGHT SUB-ARRAY
14625      IF Flag_2nd_right$(Mra_index)="Y" THEN   ! There is a sub-array
14630         IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
14635         ! Only one element is used
14640            IF (Right_2nd_array(Mra_index,1)=0) THEN
14645                 M=Right_2nd_array(Mra_index,2)
14650            ELSE
14655                 M=Right_2nd_array(Mra_index,1)
14660            END IF
14665            Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(
I))*Y_psn(M))/Lambda)
14670            FOR P=-1 TO 1 STEP 1
14675                Phase_mp=Phase_m+Theory_pha_wg(M,P)
14680                IF Theory_amp_wg(M,P)<>0 THEN
14685                     Theory_re_bp(I)=Theory_re_bp(I)+Theory_amp_wg(M,P)*COS
(Phase_mp)
14690                     Theory_im_bp(I)=Theory_im_bp(I)+Theory_amp_wg(M,P)*SIN
(Phase_mp)
14695                END IF
```

```
14700              NEXT P
14705              !
14710          ELSE  ! For more than one element in this array
14715              FOR M=Right_2nd_array(Mra_index,1) TO Right_2nd_array(Mra_inde
x,2)
14720                 Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(
Psi(I))*Y_psn(M))/Lambda)
14725                 FOR P=-1 TO 1 STEP 1
14730                    Phase_mp=Phase_m+Theory_pha_wg(M,P)
14735                    IF Theory_amp_wg(M,P)<>0 THEN
14740                       Theory_re_bp(I)=Theory_re_bp(I)+Theory_amp_wg(M,P)*CO
S(Phase_mp)
14745                       Theory_im_bp(I)=Theory_im_bp(I)+Theory_amp_wg(M,P)*SI
N(Phase_mp)
14750                    END IF
14755                 NEXT P
14760              NEXT M
14765              !
14770           END IF ! with Right_2nd_array
14775        END IF  ! for Flag_2nd_right$
14780        !
14785        !
14790     NEXT I
14795     RETURN ! THEORY_DFT
14800 ! -------------------------end-of-sub-routine--------------------------
14805 !                 .
14810  !
14815  ! ***************************************************************************
14820 Actual_dft: ! COMPUTE THE DFT OF THE ACTUAL ARRAY                            *
14825  ! ***************************************************************************
14830  !                                                                           *
14835  ! MODULE NAME : ACTUAL_DFT                    DATE: 28 SEPTEMBRE 1985     *
14840  !                                                                           *
14845  ! PROGRAMMER: SYLVAIN FLEURANT                DATE: 28 SEPTEMBER 1985     *
14850  !                                                                           *
14855  ! PURPOSE: To compute the unnormalized real and imaginary parts of a        *
14860  !          beam pattern for a specific MRA, Sensitivity, wavelength, &      *
14865  !          frequency for the actual array (with correct sensitivity).       *
14870  !                                                                           *
14875  ! INPUT: Actual_amp_wg, Actual_pha_wg, X_psn, Y_psn, Time_delay, Psi,       *
14880  !        Mra_index, freq, & lambda.                                         *
14885  ! OUTPUT: Actual_re_bp, Actual_im_bp                                        *
14890  ! SIDE EFFECTS: NIL                                                        *
14895  ! REMARKS: 1. The MRA look_up table is considered to be global i.e. it     *
14900  !          can access by this sub-routine.                                  *
14905  !                                                                           *
14910  !          2. Up to four sections can be executed, allowing up to four      *
14915  !          sub-arrays to be considered in the calculation of the beam       *
14920  !          pattern.                                                         *
14925  !                                                                           *
14930  !          3. In the present implementation, there is no left second        *
14935  !          sub-array. It is implemented but disconnected as comments.       *
14940  !          Future software development may require it uses.                 *
14945  !                                                                           *
14950  !          4. Some redundancies do exist in order to accelerate the         *
14955  !          computation speed, instead of using more sub-routine calls       *
14960  !          and context switching.                                           *
14965  ! ***************************************************************************
14970   !
14975   ! LOCAL DATA BASE DEFINITION
```

```
14980   !
14985                                   ! FALSE = Deactivated, TRUE = Activated
14990    Left_activated$="FALSE"        ! Second left sub-array deactivated
14995   !
15000   ! ------------------------------------------------------------------------
15005   !
15010   ! COMPUTE THE DFT FOR ALL PSI (Horizontal angles)
15015   !
15020    RAD             ! PSI must be in radians
15025    FOR I=-180 TO 180 STEP 1
15030      !
15035      ! ------------------------------------------------------------------
15040      ! FIRST LEFT SUB_ARRAY
15045      !
15050      FOR M=Left_1st_array(Mra_index,1) TO Left_1st_array(Mra_index,2)
15055         Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(I))
*Y_psn(M))/Lambda)
15060         FOR P=-1 TO 1 STEP 1
15065            Phase_mp=Phase_m+Actual_pha_wg(M,P)
15070            IF Actual_amp_wg(M,P)<>0 THEN
15075               Actual_re_bp(I)=Actual_re_bp(I)+Actual_amp_wg(M,P)*COS(Phase
_mp)
15080               Actual_im_bp(I)=Actual_im_bp(I)+Actual_amp_wg(M,P)*SIN(Phase
_mp)
15085            END IF
15090         NEXT P
15095      NEXT M
15100      !
15105      ! ------------------------------------------------------------------
15110      ! FIRST RIGHT SUB-ARRAY
15115      FOR M=Right_1st_array(Mra_index,1) TO Right_1st_array(Mra_index,2)
15120         !
15125         Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(I))
*Y_psn(M))/Lambda)
15130         FOR P=-1 TO 1 STEP 1
15135            Phase_mp=Phase_m+Actual_pha_wg(M,P)
15140            IF Actual_amp_wg(M,P)<>0 THEN
15145               Actual_re_bp(I)=Actual_re_bp(I)+Actual_amp_wg(M,P)*COS(Phase
_mp)
15150               Actual_im_bp(I)=Actual_im_bp(I)+Actual_amp_wg(M,P)*SIN(Phase
_mp)
15155            END IF
15160         NEXT P
15165         !
15170      NEXT M
15175      !
15180      ! ------------------------------------------------------------------
15185      ! SECOND LEFT SUB-ARRAY - DEACTIVATED FEATURES SINC NONE EXIST HERE
15190      !
15195      IF Left_activate$="TRUE " THEN
15200      IF Flag_2nd_left$(Mra_index)="Y" THEN   ! There is a sub-array
15205         IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_index,2)
=0) THEN
15210         ! Only one element is used
15215            IF (Left_2nd_array(Mra_index,1)=0) THEN
15220               M=Left_2nd_array(Mra_index,2)
15225            ELSE
15230               M=Left_2nd_array(Mra_index,1)
15235            END IF
15240            Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(
```

```
I))*Y_psn(M))/Lambda)
15245             FOR P=-1 TO 1 STEP 1
15250                Phase_mp=Phase_m+Actual_pha_wg(M,P)
15255                IF Actual_amp_wg(M,P)<>0 THEN
15260                  Actual_re_bp(I)=Actual_re_bp(I)+Actual_amp_wg(M,P)*COS(Ph
ase_mp)
15265                  Actual_im_bp(I)=Actual_im_bp(I)+Actual_amp_wg(M,P)*SIN(Ph
ase_mp)
15270                END IF
15275              NEXT P
15280             !
15285          ELSE  ! For more than one element in this array
15290              FOR M=Left_2nd_array(Mra_index,1) TO Left_2nd_array(Mra_index,
2)
15295                Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(
Psi(I))*Y_psn(M))/Lambda)
15300                FOR P=-1 TO 1 STEP 1
15305                   Phase_mp=Phase_m+Actual_pha_wg(M,P)
15310                   IF Actual_amp_wg(M,P)<>0 THEN
15315                      Actual_re_bp(I)=Actual_re_bp(I)+Actual_amp_wg(M,P)*CO
S(Phase_mp)
15320                      Actual_im_bp(I)=Actual_im_bp(I)+Actual_amp_wg(M,P)*SI
N(Phase_mp)
15325                   END IF
15330                NEXT P
15335              NEXT M
15340              !
15345           END IF ! with Left_2nd_array
15350        END IF  ! for Flag_2nd_left$
15355        END IF  ! for Left_activate$
15360        !
15365        ! ---------------------------------------------------------------
15370        ! SECOND RIGHT SUB-ARRAY
15375        IF Flag_2nd_right$(Mra_index)="Y" THEN   ! There is a sub-array
15380           IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
15385           ! Only one element is used
15390              IF (Right_2nd_array(Mra_index,1)=0) THEN
15395                 M=Right_2nd_array(Mra_index,2)
15400              ELSE
15405                 M=Right_2nd_array(Mra_index,1)
15410              END IF
15415              Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(Psi(
I))*Y_psn(M))/Lambda)
15420              FOR P=-1 TO 1 STEP 1
15425                 Phase_mp=Phase_m+Actual_pha_wg(M,P)
15430                 IF Actual_amp_wg(M,P)<>0 THEN
15435                    Actual_re_bp(I)=Actual_re_bp(I)+Actual_amp_wg(M,P)*COS
(Phase_mp)
15440                    Actual_im_bp(I)=Actual_im_bp(I)+Actual_amp_wg(M,P)*SIN
(Phase_mp)
15445                 END IF
15450              NEXT P
15455              !
15460           ELSE  ! For more than one element in this array
15465              FOR M=Right_2nd_array(Mra_index,1) TO Right_2nd_array(Mra_inde
x,2)
15470                 Phase_m=2*PI*(Freq*Time_delay(M)+(COS(Psi(I))*X_psn(M)+SIN(
Psi(I))*Y_psn(M))/Lambda)
15475                 FOR P=-1 TO 1 STEP 1
```

```
15480                   Phase_mp=Phase_m+Actual_pha_wg(M,P)
15485                   IF Actual_amp_wg(M,P)<>0 THEN
15490                      Actual_re_bp(I)=Actual_re_bp(I)+Actual_amp_wg(M,P)*CO
S(Phase_mp)
15495                      Actual_im_bp(I)=Actual_im_bp(I)+Actual_amp_wg(M,P)*SI
N(Phase_mp)
15500                   END IF
15505                NEXT P
15510             NEXT M
15515             !
15520          END IF ! with Right_2nd_array
15525       END IF  ! for Flag_2nd_right$
15530       !
15535       !
15540     NEXT I
15545     RETURN ! ACTUAL_DFT
15550  ! -------------------------end-of-sub-routine--------------------------
15555  !
15560  ! ***************************************************************************
15565 Display_input:  !  OUTPUT A TABLE OF THE INPUT DATA                          *
15570  ! ***************************************************************************
15575  !                                                                           *
15580  ! MODULE NAME: DISPLAY_INPUT                          DATE: 2 OCTOBER 1985  *
15585  !                                                                           *
15590  ! PROGRAMMER: Sylvain Fleurant                        DATE: 2 OCTOBER 1985  *
15595  !                                                                           *
15600  ! PURPOSE : Output a table of the amplitude and phase of the complex        *
15605  !           weights of both theoretical and degraded (actual) arrays,       *
15610  !           for a given MRA. Only the staves being used are shown.          *
15615  !                                                                           *
15620  ! INPUT   : Psi, Theory_pha_wg, Actual_pha_wg, Theory_amp_wg,               *
15625  !           Actual_amp_wg.                                                  *
15630  ! OUTPUT : Table of INPUT                                                   *
15635  ! CALLS  : NIL                                                             *
15640  ! SIDE EFFECTS : Table output on the printer or the CRT according to       *
15645  !                 user choice.                                              *
15650  ! REMARKS : NIL                                                            *
15655  ! ***************************************************************************
15660  !
15665  Lp_input_flag$="N"
15670   IF Output_in_data$="YES" THEN        ! OUTPUT THE ARRAY COEFFICIENTS
15675       PRINT USING "@"
15680       PRINT USING "//"
15685       PRINT "REQUEST FOR OUTPUT OF THE TABLE OF    "
15690       PRINT "THE COMPLEX RELATIVE SENSITIVITIES OF ALL "
15695       PRINT "THE HYDROPHONES USED FOR THE CHOSEN MRA"
15700       PRINT " "
15705       INPUT "DO YOU WANT THE COMPLEX SENSITIVITIES ON THE PRINTER ? (Y or
N)",Lp_input_flag$
15710       PRINT USING "@"
15715       IF Lp_input_flag$="Y" THEN
15720          PRINTER IS 701
15725       END IF
15730       PRINT
15735       PRINT "  THE REQUESTED MRA = ";Requested_mra;" degrees"
15740       PRINT "  THE USED MRA      = ";Used_mra;"  degrees"
15745       PRINT " "
15750       PRINT "  THE MODE OF OPERATION = ";Mode$
15755       PRINT
15760       PRINT "  THE COMPLEX RELATIVE SENSITIVITIES USED ARE"
```

```
15765       PRINT " "
15770       PRINT "    HYDRO.                DESIGN                    DEGRADED"
15775       PRINT "    NUMBER       AMPL X         PHASE X       AMPL X     PHAS
E X"
15780       PRINT "    M   ,  P"
15785       PRINT "-------------------------------------------------------------
----"
15790       PRINT " "
15795      !
15800      ! FIST LEFT SUB-ARRAY
15805      !
15810      FOR M=Left_1st_array(Mra_index,1) TO Left_1st_array(Mra_index,2)
15815         FOR P=-1 TO 1 STEP 1
15820           PRINT USING Formi;M;P;Theory_amp_wg(M,P);Theory_pha_wg(M,P);Actu
al_amp_wg(M,P),Actual_pha_wg(M,P)
15825         NEXT P
15830      NEXT M
15835      !
15840      ! ---------------------------------------------------------------
15845      ! FIRST RIGHT SUB-ARRAY
15850      PRINT ""
15855      FOR M=Right_1st_array(Mra_index,1) TO Right_1st_array(Mra_index,2)
15860         !
15865         FOR P=-1 TO 1 STEP 1
15870           PRINT USING Formi;M;P;Theory_amp_wg(M,P);Theory_pha_wg(M,P);Actu
al_amp_wg(M,P),Actual_pha_wg(M,P)
15875         NEXT P
15880         !
15885      NEXT M
15890      !
15895      ! ---------------------------------------------------------------
15900      ! SECOND LEFT SUB-ARRAY - DEACTIVATED FEATURES SINC NONE EXIST HERE
15905      !
15910      IF Left_activate$="TRUE " THEN
15915      PRINT " SECOND LEFT SUB-ARRAY"
15920      IF Flag_2nd_left$(Mra_index)="Y" THEN   ! There is a sub-array
15925         IF (Left_2nd_array(Mra_index,1)=0) OR (Left_2nd_array(Mra_index,2)
=0) THEN
15930         ! Only one element is used
15935            IF (Left_2nd_array(Mra_index,1)=0) THEN
15940                 M=Left_2nd_array(Mra_index,2)
15945            ELSE
15950                 M=Left_2nd_array(Mra_index,1)
15955            END IF
15960            FOR P=-1 TO 1 STEP 1
15965                 PRINT USING Formi;M;P;Theory_amp_wg(M,P);Theory_pha_wg(M,P
);Actual_amp_wg(M,P),Actual_pha_wg(M,P)
15970             NEXT P
15975            !
15980         ELSE  ! For more than one element in this array
15985            FOR M=Left_2nd_array(Mra_index,1) TO Left_2nd_array(Mra_index,
2)
15990               FOR P=-1 TO 1 STEP 1
15995                    PRINT USING Formi;M;P;Theory_amp_wg(M,P);Theory_pha_wg
(M,P);Actual_amp_wg(M,P),Actual_pha_wg(M,P)
16000               NEXT P
16005            NEXT M
16010            !
16015         END IF ! with Left_2nd_array
16020      END IF  ! for Flag_2nd_left$
```

```
16025       END IF  ! for Left_activate$
16030       !
16035       ! ------------------------------------------------------------------
16040       ! SECOND RIGHT SUB-ARRAY
16045       IF Flag_2nd_right$(Mra_index)="Y" THEN   ! There is a sub-array
16050          PRINT " SECOND RIGHT SUB-ARRAY"
16055          IF (Right_2nd_array(Mra_index,1)=0) OR (Right_2nd_array(Mra_index,
2)=0) THEN
16060          ! Only one element is used
16065             IF (Right_2nd_array(Mra_index,1)=0) THEN
16070                M=Right_2nd_array(Mra_index,2)
16075             ELSE
16080                M=Right_2nd_array(Mra_index,1)
16085             END IF
16090             FOR P=-1 TO 1 STEP 1
16095                   PRINT USING Formi;M;P;Theory_amp_wg(M,P);Theory_pha_wg
(M,P);Actual_amp_wg(M,P),Actual_pha_wg(M,P)
16100             NEXT P
16105             !
16110          ELSE  ! For more than one element in this array
16115             FOR M=Right_2nd_array(Mra_index,1) TO Right_2nd_array(Mra_inde
x,2)
16120                FOR P=-1 TO 1 STEP 1
16125                   PRINT USING Formi;M;P;Theory_amp_wg(M,P);Theory_pha_wg
(M,P);Actual_amp_wg(M,P),Actual_pha_wg(M,P)
16130                NEXT P
16135             NEXT M
16140             !
16145          END IF ! with Right_2nd_array
16150       END IF  ! for Flag_2nd_right$
16155       !
16160       !
16165       ! ------------------------------------------------------------------
16170       !
16175       PRINT USING "@"                ! GO TO THE END OF THE PAGE
16180       IF Lp_input_flag$="Y" THEN
16185          Lp_input_flag$="N"
16190          PRINTER IS 1
16195       END IF
16200    END IF
16205 Formi:    IMAGE 3X,DDD,2X,DD,5X,3D.5D,5X,3D.5D,5X,3D.5D,3X,3D.5D
16210  RETURN
16215  !
16220  ! -------------------------end-of-sub-routine-------------------------
16225  !
16230  ! ***************************************************************************
16235 Display_output:  !   OUTPUT A TABLE OF THE OUTPUT DATA                       *
16240  ! ***************************************************************************
16245  !                                                                           *
16250  ! MODULE NAME: DISPLAY_OUTPUT                          DATE: 24 MAY 1985    *
16255  !                                                                           *
16260  ! PURPOSE : Output a table of normalized magnitude and magnitude in dB      *
16265  !           of the far-field beam pattern for both theoretical and          *
16270  !           degraded (or actual) cases as function of the horizontal        *
16275  !           angle (degrees).                                                *
16280  !                                                                           *
16285  ! INPUT   : Psi, Theory_mag_bp, Theory_db_bp, Actual_mag_bp,                *
16290  !           Actual_db_bp, Insert_line$, Temporary_real, Lp_input_flag$,     *
16295  !           Output_bp_data$, Ok_input_flag$, Mode$, Requested_mra,          *
16300  !           Used_mra, T1, T2 .                                              *
```

```
16305  ! OUTPUT : Table of Psi, Theory_mag_bp, Theory_db_bo, Actual_mag_bp,    *
16310  !          Actual_db_bp                                                 *
16315  ! CALLS  : NIL                                                         *
16320  ! SIDE EFFECTS : Table output on the printer or the CRT according to   *
16325  !                 user choice.                                         *
16330  ! REMARKS : NIL                                                        *
16335  ! ***********************************************************************
16340  !
16345  !
16350  ! DEFINE LOCAL DATA BASE
16355  !
16360    INTEGER Out_angle_steps          ! Step used for the output of the BP
16365                                     ! It is in degrees
16370                                     !
16375    Out_angle_steps=1                ! Every one degrees is standard.
16380                                     !
16385  Lp_input_flag$="N"
16390  IF Output_bp_data$="YES" THEN     ! OUTPUT THE COMPUTED BEAM PATTERN
16395       PRINT USING "@"
16400       PRINT "THE TABLE OF THE NORMALIZED MAGNITUDE "
16405       PRINT "RECTANGULAR AND dB FORMATS"
16410       PRINT "OF THE  BEAM PATTERN CURVES WILL BE DISPLAYED"
16415       PRINT "AS FUNCTION OF THE HORIZONTAL ANGLE"
16420       PRINT "BOTH THEORETICAL AND DEGRADED CONDITION CASE"
16425       PRINT " "
16430       PRINT "PLEASE, PROVIDE THE INCREMENTAL STEPS FOR THE ANGLE"
16435       PRINT "FOR EXAMPLE, EVERY 1 OR 5 OR 10 DEGREES."
16440       PRINT " "
16445       !
16450       ! ---------------------------------------------------------------
16455       !
16460       ! Insert the step used for the angles
16465       Insert_line$=" "      ! Initialize to nothing
16470         ! MAKE SURE THAT ALL INPUT DATA ARE OF THE CORRECT TYPE
16475       REPEAT
16480          INPUT "ENTER THE STEP FOR THE ANGLES IN DEGREES",Insert_line$
16485             ! Check if the string is positive numeric - Pass by value
16490          Ok_input_flag$=FNIs_string_pnum$((Insert_line$))
16495          IF Ok_input_flag$="TRUE " THEN
16500               ! This is a numeric string
16505               ! Enter the first real number in the string
16510               ! skip all the rest if any characters are left
16515               ENTER Insert_line$;Temporary_real  ! Define in User_menu
16520               Out_angle_steps=INT(Temporary_real)
16525               IF (Out_angle_steps<1) THEN
16530                   DISP " YOUR NUMBER IS NOT PHYSICALLY POSSIBLE"
16535                   WAIT 2.0    ! Wait 2 sec
16540                   Ok_input_flag$="FALSE"
16545               END IF
16550          ELSE
16555               ! This is not a POSITIVE number
16560               DISP " YOU MUST ENTER A POSITIVE NUMBER"
16565               WAIT .75   ! Wait 3/4 sec so user can read it
16570         END IF    ! of Ok_input_flag$ is TRUE
16575          UNTIL Ok_input_flag$="TRUE "    ! i.e. a good frequency
16580          !
16585       !
16590       ! ---------------------------------------------------------------
16595       !
16600       INPUT "DO YOU WANT THE BEAM PATTERN DATA ON THE PRINTER ? (Y or N)",
```

```
Lp_input_flag$
16605       IF Lp_input_flag$="Y" THEN
16610          PRINTER IS 701
16615       END IF
16620       !
16625       ! ------------------------------------------------------------------
16630       !
16635       PRINT USING "@"
16640       PRINT
16645       PRINT "  THE REQUESTED MRA = ";Requested_mra;" degrees"
16650       PRINT "  THE USED MRA      = ";Used_mra;"  degrees"
16655       PRINT " "
16660       PRINT "  THE MODE OF OPERATION = ";Mode$
16665       PRINT " "
16670       PRINT "  The execution time to compute the beam pattern was ";TZ-T1;
" SEC"
16675       PRINT " "
16680       PRINT "       *** COMPUTED MAGNITUDE BEAM PATTERN DATA ***"
16685       PRINT " "
16690       PRINT "    ANGLE          THEORETICAL                    ACTUAL"
16695       PRINT "   degrees.   Normalized       dB          Normalized       d
B"
16700       PRINT "  ------------------------------------------------------------
---"
16705       PRINT USING "/"
16710       FOR I=-180 TO 180 STEP Out_angle_steps   ! From/to +/- 180 degrees
16715          PRINT USING Formo;Psi(I);Theory_mag_bp(I);Theory_db_bp(I);Actual_
mag_bp(I);Actual_db_bp(I)
16720       NEXT I
16725       PRINT " "
16730        ! PRINT USING "@"            ! GO TO THE END OF THE PAGE
16735       IF Lp_input_flag$="Y" THEN
16740          Lp_input_flag$="N"
16745          PRINTER IS 1
16750       END IF
16755        !
16760    END IF
16765       !
16770 Formo:   IMAGE 3X,DDDDD,5X,3D.5D,7X,3D.2D,7X,3D.5D,7X,3D.2D
16775       !
16780  RETURN
16785  ! ------------------------end-of-sub-routine--------------------------
16790  !
16795  ! ***************************************************************************
16800 Graph_output:  ! OUTPUT THE GRAPHS PREVIOUSLY REQUESTED WITH MENU          *
16805  ! ***************************************************************************
16810  !                                                                          *
16815  ! MODULE NAME: GRAPH_OUTPUT                          DATE: 4 AUGUST 1985  *
16820  !                                                                          *
16825  ! PURPOSE:  Provide the proper graph that the user previously requested *
16830  !           on the main menu. Provide him with a choice of medium on     *
16835  !           which a given graph can be transfered to (if desired).       *
16840  !                                                                        *
16845  ! INPUT  : Graph_db_flag$, Graph_ma_flag$, Graph_po_flag$, and           *
16850  !          Graph_op_medium$                                              *
16855  ! OUTPUT : NIL                                                          *
16860  ! CALLS  : NIL                                                         *
16865  ! SIDE EFFECTS : NIL                                                   *
16870  ! REMARKS : Refer to the main DATABASE definition for the valid value   *
16875  !           of Graph_op_medium$.                                        *
```

```
16880  !                                                                      *
16885  ! ***********************************************************************
16890  !    Graph_op_medium$[4]       ! To indicate the type of output medium
16895                                 ! Definition:  CRT_ = CRT Output
16900                                 ! PRHN = Printer HP2671G Normal size
16905                                 ! PRTN = Printer Thinkjet Normal size
16910                                 ! PRHE = Printer HP2671G Expanded size
16915                                 ! PRTE = Printer Thinkjet Expanded size
16920                                 ! PLOT = Plotter Normal size
16925  !
16930  OUTPUT 2;Clear$;                    ! Clear the CRT
16935  IF (Graph_db_flag$="NO ") AND (Graph_ma_flag$="NO ") AND (Graph_po_flag$
="NO ") THEN
16940       ! NO GRAPH WILL BE GENERATED - INFORM THE USER ABOUT IT
16945       ALPHA OFF
16950       GRAPHICS ON          ! Use large lettering to inform the user
16955       GINIT
16960       PEN 3
16965       LORG 5
16970       CSIZE 6
16975       MOVE,66,65
16980       LABEL "AS REQUESTED"
16985       MOVE 66,55
16990       LABEL "NO GRAPHS WILL BE GENERATED"
16995       WAIT 5                        ! Wait 5 secondes - so user can read
17000       GRAPHICS OFF
17005       ALPHA ON                      ! Restore normal condition
17010       OUTPUT 2;Clear$;
17015  !
17020  ELSE   ! At least one graph is requested
17025       !
17030       !
17035       IF Graph_db_flag$="YES" THEN
17040            GOSUB Plot_db_bp
17045            WAIT 1             ! Allow the user to quickly see the graph
17050            GOSUB Gph_size_option    ! Choose on what medium to output
17055            GOSUB Graph_info         ! Show the procedure to follow
17060            IF (Graph_op_medium$="CRT_") THEN
17065                 OFF KEY
17070                 ALPHA OFF
17075                 GRAPHICS ON         ! Just show the already drawn graph
17080                 PAUSE
17085            ELSE
17090                 ! It is going to be printed or plotted
17095               IF (Graph_op_medium$="PRHN") OR (Graph_op_medium$="PRTN") T
HEN
17100                 ALPHA OFF
17105                 GRAPHICS ON
17110                 DUMP GRAPHICS CRT TO #701    ! Dump unto the printer
17115                 GRAPHICS OFF
17120                 ALPHA ON
17125                 PAUSE
17130               ELSE
17135                 IF (Graph_op_medium$="PRHE") OR (Graph_op_medium$="PRTE"
) THEN
17140                   GOSUB Plot_db_bp
17145                   DUMP DEVICE IS 701,EXPANDED  ! Size is doubled
17150                   DUMP GRAPHICS
17155                   Graph_scale=1.0   ! Restore the graph scale
17160                   PAUSE
```

```
17165                ELSE
17170                   Do_plot$="TRUE "
17175                   GOSUB Plot_db_bp
17180                   Do_plot$="FALSE"
17185                   PLOTTER IS CRT,"INTERNAL"
17190                   PAUSE
17195                END IF ! Print EXPANDED
17200             END IF ! Normal print out
17205          END IF ! CRT output
17210       END IF   ! yes for dB graph
17215       !
17220       ! ---------------------------------------------------------------
17225       !
17230       IF Graph_ma_flag$="YES" THEN
17235          GOSUB Plot_magn_bp
17240          WAIT 1              ! Allow the user to quickly see the graph
17245          GOSUB Gph_size_option    ! Choose on what medium to output
17250          GOSUB Graph_info         ! Show the procedure to follow
17255          IF (Graph_op_medium$="CRT_") THEN
17260                OFF KEY
17265                ALPHA OFF
17270                GRAPHICS ON        ! Just show the already drawn graph
17275                PAUSE
17280          ELSE
17285                ! It is going to be printed or plotted
17290             IF (Graph_op_medium$="PRHN") OR (Graph_op_medium$="PRTN") T
HEN
17295                ALPHA OFF
17300                GRAPHICS ON
17305                DUMP GRAPHICS CRT TO #701     ! Dump unto the printer
17310                GRAPHICS OFF
17315                ALPHA ON
17320                PAUSE
17325             ELSE
17330                IF (Graph_op_medium$="PRHE") OR (Graph_op_medium$="PRTE"
) THEN
17335                   GOSUB Plot_magn_bp
17340                   DUMP DEVICE IS 701,EXPANDED  ! Size is doubled
17345                   DUMP GRAPHICS
17350                   Graph_scale=1.0   !Restore the graph scale
17355                   PAUSE
17360                ELSE
17365                   Do_plot$="TRUE "
17370                   GOSUB Plot_magn_bp
17375                   Do_plot$="FALSE"
17380                   PLOTTER IS CRT,"INTERNAL"
17385                   PAUSE
17390                END IF ! Print EXPANDED
17395             END IF ! Normal print out
17400          END IF ! CRT output
17405       END IF   ! Yes for ma graph
17410       !
17415       ! ---------------------------------------------------------------
17420       !
17425       IF Graph_po_flag$="YES" THEN
17430          GOSUB Plot_po_bp
17435          WAIT 2              ! Allow the user to quickly see the graph
17440          GOSUB Gph_size_option    ! Choose on what medium to output
17445          GOSUB Graph_info         ! Show the procedure to follow
17450          IF (Graph_op_medium$="CRT_") THEN
```

```
17455                 OFF KEY
17460                 ALPHA OFF
17465                 GRAPHICS ON        ! Just show the already drawn graph
17470                 PAUSE
17475           ELSE
17480                 ! It is going to be printed or plotted
17485              IF (Graph_op_medium$="PRHN") OR (Graph_op_medium$="PRTN") T
HEN
17490                 ALPHA OFF
17495                 GRAPHICS ON
17500                 DUMP GRAPHICS CRT TO #701    ! Dump unto the printer
17505                 GRAPHICS OFF
17510                 ALPHA ON
17515                 PAUSE
17520              ELSE
17525                 IF (Graph_op_medium$="PRHE") OR (Graph_op_medium$="PRTE"
) THEN
17530                    GOSUB Plot_po_bp
17535                    DUMP DEVICE IS 701,EXPANDED  ! Size is doubled
17540                    DUMP GRAPHICS
17545                    Graph_scale=1.0   ! Restore the graph scale
17550                    PAUSE
17555                 ELSE
17560                    Do_plot$="TRUE "
17565                    GOSUB Plot_po_bp
17570                    Do_plot$="FALSE"
17575                    PLOTTER IS CRT,"INTERNAL"
17580                    PAUSE
17585                 END IF ! Print EXPANDED
17590              END IF ! Normal print out
17595           END IF ! CRT output
17600           !
17605        END IF   ! Yes for polar graph
17610        !
17615        ! ---------------------------------------------------------------
17620        !
17625     !
17630     END IF ! NO GRAPHS
17635    RETURN ! GRAPH_OUTPUT
17640  !
17645  ! -------------------------end-of-sub-routine--------------------------
17650  !
17655  ! ****************************************************************************
17660 Graph_info:! PROVIDE THE USER WITH INFORMATION ABOUT HOW TO PROCEED       *
17665  ! ****************************************************************************
17670  !                                                                           *
17675  ! MODULE NAME: GRAPH_INFO                              DATE: 6 AUGUST 1985 *
17680  !                                                                           *
17685  ! PURPOSE:  Provide the user with information about how to proceed       *
17690  !           when the graph has been plotted/printed/examined, i.e.        *
17695  !           how to goto the next graph or feature. Basically, it is       *
17700  !           done with the key CONTINUE.                                   *
17705  !                                                                         *
17710  !           selection to another on the menu.                             *
17715  ! INPUT  : NIL                                                           *
17720  ! OUTPUT : NIL                                                           *
17725  ! CALLS  : NIL                                                          *
17730  ! SIDE EFFECTS : Clear the screen and print a message.                   *
17735  !                 The system is then pause.                             *
17740  ! REMARKS : NIL                                                         *
```

```
17745  ! ************************************************************************
17750  !
17755      PRINT USING "@"           ! Clear the alpha portion of the graph.
17760      PRINT TABXY(4,15);"      Your request will be executed."
17765      PRINT USING "//"
17770      PRINT TABXY(4,20);"           When you are done,"
17775      PRINT USING "//"
17780      PRINT TABXY(4,25);"Please press CONTINUE to go to the next feature"
17785      PRINT USING "/"
17790      PRINT TABXY(4,30);" After you have read this , press CONTINUE"
17795      PAUSE   ! WAIT UNTIL THE USER HAD READ IT
17800    !
17805    RETURN ! Graph_info
17810  ! -------------------------end-of-sub-routine--------------------------
17815  !
17820  ! ************************************************************************
17825 Gph_size_option:! PROVIDE A CHOICE OF OUTPUT MEDIUM FOR A GRAPH          *
17830  ! ************************************************************************
17835  !                                                                       *
17840  ! MODULE NAME: GRP_SIZE_OPTION                    DATE: 2 AUGUST 1985 *
17845  !                                                                       *
17850  ! PURPOSE:  Provide a choice to the user of the different output medium *
17855  !           on which a graph can be generated. If no hard copy output,  *
17860  !           is required the CRT is used.                                *
17865  !                                                                       *
17870  ! INPUT  : Graph_scale, Graph_op_medium                                 *
17875  ! OUTPUT : Graph_scale, Graph_op_medium                                 *
17880  ! CALLS  : NIL                                                         *
17885  ! SIDE EFFECTS : NIL                                                   *
17890  ! REMARKS : Refer to the main DATABASE definition for the valid value  *
17895  !           of Graph_op_medium.                                         *
17900  !           Use KNOB or KEY 3 and KEY 4 to move the marker from one     *
17905  !           selection to another on the menu.                           *
17910  ! ************************************************************************
17915  !
17920  ! LOCAL DATA BASE DEFINITION
17925  !
17930    DIM Marker$[4]                    ! Use as pointer arrow
17935    DIM Toggle$[8]                    ! Use to alternate view Graphics/Menu
17940    INTEGER Point                     ! To keep track of arrow position
17945    INTEGER Max_selection             ! Maximum number of choices
17950    INTEGER Vert_offset               ! Number of lines to skip
17955  !
17960    GRAPHICS OFF
17965    ALPHA ON
17970    CONTROL 1,4;0             ! Display function OFF, if not already done
17975    CONTROL 2,1;0             ! PRINTER ALL OFF, if not already done
17980    PRINT USING "@"           ! Just clear the alpha portion of the screen
17985    !
17990    Toggle$="MENU    "
17995    Vert_offset=5
18000    Max_selection=6           ! Only 6 selections are available.
18005    Point=1                   ! Default selection is the CRT
18010    Marker$="=>"&CHR$(8)&CHR$(8)   ! Define the arrow maker
18015    !
18020    PRINT
18025    PRINT " Use shift and knob to move marker to a selection"
18030    PRINT "     MEDIUM TYPE              SIZE"
18035    PRINT
18040    PRINT "     CRT ONLY"
```

```
18045    PRINT "     PRINTER HP2671G         NORMAL  (HALF PAGE)"
18050    PRINT "     PRINTER HP2671G        EXPANDED (FULL PAGE)"
18055    PRINT "     PRINTER THINKJET        NORMAL  (HALF PAGE)"
18060    PRINT "     PRINTER THINKJET       EXPANDED (FULL PAGE)"
18065    PRINT "     PLOTTER                 NORMAL  (FULL PAGE)"
18070    PRINT USING "/"
18075    PRINT " Use KNOB or KEY 3 and KEY 4 to move the marker"
18080    PRINT " Press KEY 0 to see your graph again and back"
18085    PRINT " Press KEY 9 when a selection is made"
18090    !
18095    PRINT TABXY(4,Point+Vert_offset);Marker$; ! Display the starting marker
18100    !
18105    ON KNOB .6 GOTO Move_pointerlbl      ! Enable the knob
18110    !
18115    ! ------------------------------------------------------------------------
18120    !
18125    ! DEFINE KEY CHOICE
18130    !
18135    CONTROL 1,12;0              ! Activate the softkeys and its display
18140    ON KEY 9 LABEL "ENTER CHOICE" GOTO Enter_choice
18145    ON KEY 0 LABEL " TOGGLE " GOTO Toggle
18150    ON KEY 1 GOTO Not_used_lbl
18155    ON KEY 2 GOTO Not_used_lbl
18160    ON KEY 3 LABEL " V (DOWN)" GOTO Gph_option_down
18165    ON KEY 4 LABEL " ^ (UP)" GOTO Gph_option_up
18170    ON KEY 5 GOTO Not_used_lbl
18175    ON KEY 6 GOTO Not_used_lbl
18180    ON KEY 7 GOTO Not_used_lbl
18185    ON KEY 8 GOTO Not_used_lbl
18190    !
18195    ! ------------------------------------------------------------------------
18200    !
18205 Spin_wheel:   GOTO Spin_wheel          ! Wait for knob interrupt
18210                                        ! or for KEY interrupt.
18215    !
18220    ! ------------------------------------------------------------------------
18225    !
18230 Move_pointerlbl:                       ! Check for direction
18235    IF KNOBY>0 THEN
18240       Point=Point+1
18245    ELSE
18250       Point=Point-1
18255    END IF
18260    IF Point<1 THEN Point=Max_selection ! Keep pointer within limits
18265    IF Point>Max_selection THEN Point=1
18270    PRINT "  ";                         ! Erase the old marker
18275    PRINT TABXY(4,Point+Vert_offset);Marker$;    ! Display the new marker
18280    GOTO Spin_wheel
18285    !
18290    ! ------------------------------------------------------------------------
18295    !
18300 Gph_option_up:       ! Go up one more selection in the MENU
18305    Point=Point-1      ! Add one more to the locator
18310    IF Point<1 THEN Point=Max_selection ! Keep pointer within limits
18315    IF Point>Max_selection THEN Point=1   ! Reset to the first one (scroll)
18320    PRINT "  ";                         ! Erase the old marker
18325    PRINT TABXY(4,Point+Vert_offset);Marker$;   ! Display the new marker
18330    GOTO Spin_wheel
18335    !
18340    ! ------------------------------------------------------------------------
```

```
18345    !
18350 Gph_option_down:      ! Go down one more selection in the MENU
18355    Point=Point+1      ! Subtract one more to the locator
18360    IF Point<1 THEN Point=Max_selection  ! keep pointer within limits
18365    IF Point>Max_selection THEN Point=1   ! Reset to the first one (scroll)
18370    PRINT "  ";                          ! Erase the old marker
18375    PRINT TABXY(4,Point+Vert_offset);Marker$;   ! Display the new marker
18380    GOTO Spin_wheel
18385    !
18390    ! ------------------------------------------------------------------
18395    !
18400    !
18405 Toggle: ! Alternate view between graphs and menu
18410       IF Toggle$="MENU    " THEN
18415           Toggle$="GRAPHICS"
18420           GRAPHICS ON
18425           ALPHA OFF
18430       ELSE
18435           Toggle$="MENU    "
18440           GRAPHICS OFF
18445           ALPHA ON
18450       END IF
18455       GOTO Spin_wheel
18460    !
18465    ! ------------------------------------------------------------------
18470    !
18475 Not_used_lbl:                 ! The choosen key is undefined
18480       BEEP 500,.3             ! Prompt the user
18485       GOTO Move_pointerlbl
18490    !
18495    ! ------------------------------------------------------------------
18500    !
18505 Enter_choice:   ! Enter the choice made
18510    !
18515    Graph_scale=1               ! Standard default is the maximum scale
18520    IF Point=1 THEN
18525         Graph_op_medium$="CRT_"
18530    ELSE
18535         IF Point=2 THEN
18540             Graph_op_medium$="PRHN"   ! For HP2671G Normal graph
18545                                       ! or any similar printer
18550         ELSE
18555             IF Point=3 THEN
18560                 Graph_op_medium$="PRHE"
18565                 Graph_scale=.698            ! For HP2671G Expanded graph
18570             ELSE
18575                 IF Point=4 THEN
18580                     Graph_op_medium$="PRTN"  ! For Thinkjet Normal graph
18585                 ELSE
18590                     IF Point=5 THEN
18595                         Graph_op_medium$="PRTE"
18600                         Graph_scale=.750   ! For Thinkjet Expanded graph
18605                     ELSE
18610                         Graph_op_medium$="PLOT"  ! For Plotter
18615                     END IF  ! Of point = 5
18620                 END IF   ! Of point = 4
18625             END IF   ! Of point = 3
18630         END IF  ! Of point = 2
18635    END IF ! Of point = 1
18640    PRINT TABXY(1,22),"THE GRAPH TYPE IS ";Graph_op_medium$
```

```
18645    WAIT .5
18650    !
18655    CONTROL 1,12;1      ! Turn off the softkeys and their display.
18660    !                    -
18665    !
18670  RETURN ! OF GPH_SIZE_OPTION
18675  ! -------------------------end-of-sub-routine--------------------------
18680  !
18685  ! ****************************************************************************
18690 Insert_bot_line:   !  INSERT LINE AT THE BOTTOM OF THE GRAPH                  *
18695  ! ****************************************************************************
18700  !                                                                            *
18705  ! MODULE NAME: INSERT_BOT_LINE                          DATE: 27 JUNE 1985  *
18710  !                                                                            *
18715  ! PURPOSE : Insert at the bottom of the graph of the beam pattern a          *
18720  !           line of information indicating the date, spacing between         *
18725  !           elements, frequency, speed, and medium used.                     *
18730  !                                                                            *
18735  ! INPUT   : TIMEDATE, Frequency, Spacing, Speed, Medium, Graph_scale        *
18740  ! OUTPUT : Line at bottom of graph                                          *
18745  ! CALLS  : ALL PLOTTING SUBROUTINES                                         *
18750  ! SIDE EFFECTS : NIL                                                        *
18755  ! REMARKS : NIL                                                            *
18760  !                                                                            *
18765  ! ****************************************************************************
18770      CSIZE INT(Graph_scale*3)              ! smaller lettering
18775      MOVE 2,2
18780      LORG 2
18785       LABEL DATE$(TIMEDATE)                ! output date
18790      LORG 8
18795       LABEL "   "                          ! any relevant information
18800      MOVE 35,2
18805      LORG 5
18810      ! LABEL "Medium: ";Medium$            ! indicate medium being used
18815       LABEL "MODE: ";Mode$                 ! indicate the requested MODE
18820      MOVE 63,2
18825       LABEL "Frequency: ";Freq;" Hz"
18830      MOVE 105,2
18835       LABEL "Speed: ";Speed;" m/s"         ! indicate the speed of sound
18840      !
18845      ! DRAW A STRAIGHT LINE ALONG X TO FORM A BOX
18850      !
18855        MOVE 0,4
18860        DRAW 133,4
18865      !
18870      RETURN   ! of Insert_bot_line
18875  ! -------------------------end-of-sub-routine--------------------------
18880  !
18885  ! ****************************************************************************
18890 Insert_legend: ! INSERT LEGEND ON THE SIDE OF THE GRAPH                       *
18895  ! ****************************************************************************
18900  !                                                                            *
18905  ! MODULE NAME: INSERT LEGEND                        DATE: 3 OCTOBER 1985    *
18910  !                                                                            *
18915  ! PURPOSE : Insert the legend for the graph on the side of the graph        *
18920  !           for the design and the degraded beam pattern.                    *
18925  !                                                                            *
18930  ! INPUT  : Graph_scale, Line_type_theo, Line_type_actl                       *
18935  ! OUTPUT : Line on the side of graph                                        *
18940  ! CALLS  : ALL PLOTTING SUBROUTINES                                         *
```

```
18945  ! SIDE EFFECTS : NIL                                                    *
18950  ! REMARKS : NIL                                                        *
18955  !                                                                      *
18960  ! ************************************************************************
18965      CSIZE INT(Graph_scale*3)              ! smaller lettering
18970      MOVE 125,65
18975      LORG 5
18980      LABEL "LEGEND"
18985      CSIZE INT(Graph_scale*2)              ! Even smaller lettering
18990      LINE TYPE Line_type_theo
18995      MOVE 120,60
19000      DRAW 130,60
19005      MOVE 125,58
19010       LORG 6
19015       LINE TYPE 1
19020       LABEL "DESIGN"                       ! Output 1st curve type
19025      LINE TYPE Line_type_actl
19030      MOVE 120,52
19035      DRAW 130,52
19040      MOVE 125,50
19045       LINE TYPE 1
19050       LABEL "DEGRADED"                     ! Output 2nd curve type
19055                                            !
19060      MOVE 133,46                           ! DRAW A BOX AROUND THE LEGEND
19065      DRAW 118,46
19070      DRAW 118,70
19075      DRAW 133,70
19080      !
19085    RETURN ! Insert_legend
19090  ! ---------------------------end-of-sub-routine---------------------------
19095  !
19100  ! ************************************************************************
19105 Plot_magn_bp:!      PLOT NORMALIZED MAGNITUDE OF THE BEAM PATTERN        *
19110  ! ************************************************************************
19115  !                                                                        *
19120  ! MODULE NAME: PLOT_MAGN_BP                    DATE: 11 SEPTEMBER 1985  *
19125  !                                                                        *
19130  ! PURPOSE : Plot the graph of the magnitude of the beam pattern versus  *
19135  !           Horizontal angle (degrees).                                  *
19140  !                                                                        *
19145  ! INPUT  :  Do_plot$, Graph_scale, Used_mra, Psi(*), Theory_mag_bp,      *
19150  !           Actual_mag_bp, Line_type_theo, Line_type_actl.               *
19155  ! OUTPUT : Graph of Psi VERSUS the Magnitude of the beam pattern         *
19160  ! CALLS  : Graph_mag_label, Insert_bot_line                              *
19165  ! SIDE EFFECTS : NIL                                                    *
19170  ! REMARKS : 1. PEN 1 : EVERYTHING EXCEPT TITLE AND CURVES               *
19175  !              PEN 2 : FOR TITLE                                        *
19180  !              PEN 3 : FOR CURVES                                       *
19185  !                                                                       *
19190  ! ************************************************************************
19195    !
19200    GOSUB Graph_mag_label                 ! Labels plot
19205    OUTPUT 2;Clear$;                      ! Clear Screen
19210    ALPHA OFF
19215    GCLEAR                                ! Clear Graphics Screen
19220    GRAPHICS ON
19225    GINIT                                 ! Initialize default graphics
19230    IF Do_plot$="TRUE " THEN
19235        PLOTTER IS 705,"HPGL"
19240    END IF
```

```
19245     DEG                                    ! All angles are in degrees
19250     !
19255     ! ------------------------------------------------------------------
19260     !
19265     PEN 1
19270     VIEWPORT 0,INT(Graph_scale*133),0,INT(Graph_scale*100)
19275     FRAME                                  ! Main frame is X = 0 TO 133
19280                                            !         AND    Y = 0 TO 100
19285     !     LABEL THE GRAPH
19290     !
19295     PEN 2
19300     LORG 6                                 ! or any positive pen
19305     MOVE 67,98
19310     CSIZE INT(Graph_scale*5.3)
19315     LABEL "MAG OF HORIZONTAL BEAM PATTERN"   ! MAIN TITLE
19320     MOVE 67,91
19325     CSIZE INT(Graph_scale*4)
19330     PEN 1
19335     LABEL "MRA : ";Used_mra;" degrees"     ! SUB TITLE
19340     MOVE 3,50
19345     LDIR 90                                ! rotate 90 Degrees
19350     LABEL "BEAM PATTERN"                   ! Y AXIS TITLE (ctd)
19355     LDIR 0                                 ! back to 0 Degree
19360     !
19365     ! ------------------------------------------------------------------
19370     !
19375     GOSUB Insert_bot_line   ! Insert line of info at the bottom of graph
19380     !
19385     ! ------------------------------------------------------------------
19390     !
19395     GOSUB Insert_legend     ! Insert the legend for this graph
19400     ! ------------------------------------------------------------------
19405     !
19410     !           LABEL THE X AXIS
19415     LORG 5
19420     MOVE 65,7
19425         LABEL "HORIZONTAL ANGLE (Degrees)"
19430     ! ------------------------------------------------------------------
19435     !
19440     ! CREATE THE FRAME WHICH WILL CONTAINS THE ACTUAL GRAPH
19445     !
19450     VIEWPORT INT(Graph_scale*15),INT(Graph_scale*115),INT(Graph_scale*12),
INT(Graph_scale*87)
19455     WINDOW Left,Right,Bottom,Top
19460     AXES Xsp,Ysp,Left,Right,Xmaj,Ymaj,Majsz     ! draw axes LEFT & RIGHT
19465     AXES Xsp,Ysp,Right,Top,Xmaj,Ymaj,Majsz      ! draw axes TOP & RIGHT
19470     AXES Xsp,Ysp,0,Bottom,Xmaj,Ymaj,Majsz       ! draw axes LEFT & BOTTOM
19475     !
19480     LINE TYPE 4                            ! dotted line
19485     !   No grid line are outputed if a plot is done
19490     IF Do_plot$="FALSE" THEN
19495        GRID Xgrid,Ygrid
19500     END IF
19505     LINE TYPE 1                            ! continuous line
19510     FRAME
19515     LORG 8
19520     CSIZE INT(Graph_scale*3)
19525     CLIP OFF
19530     !
19535     FOR L=Bottom TO Top STEP Ystepsz       ! indicate Y steps
```

```
19540          MOVE Left,L
19545          IF ABS(L)<1.E-6 THEN
19550             LABEL "0 "
19555          ELSE
19560             LABEL L
19565          END IF
19570       NEXT L  !
19575       !
19580       ! ------------------------------------------------------------------------
19585       !
19590       !   OUTPUT THE X DIVISIONS FOR HORIZONTAL ANGLES
19595       !
19600       LORG 6
19605       FOR L=Left TO Right STEP (Xstepsz*4)
19610             MOVE L,Bottom
19615              LABEL L
19620       NEXT L
19625       !
19630       ! ------------------------------------------------------------------------
19635       !
19640       !    DRAW THE CURVE
19645       !
19650       CSIZE INT(Graph_scale*5)
19655       CLIP ON                              ! Keep the curves inside the window
19660       !
19665       !*****************************************************
19670       ! THEORETICAL CURVE
19675       LINE TYPE Line_type_theo
19680       PEN 3
19685       MOVE Psi(-180),Theory_mag_bp(-180)   ! IMPORTANT, RETURN TO THE 1ST PT
19690       FOR J=-180 TO 180 STEP 1
19695             DRAW Psi(J),Theory_mag_bp(J)
19700       NEXT J
19705       !
19710       !*****************************************************
19715       ! ACTUAL OR DEGRADED CURVE
19720       LINE TYPE Line_type_actl
19725       PEN 3
19730       MOVE Psi(-180),Actual_mag_bp(-180)
19735       FOR K=-180 TO 180 STEP 1
19740            DRAW Psi(K),Actual_mag_bp(K)
19745       NEXT K
19750       !
19755       LDIR 0
19760       PEN 0                                ! PEN UP
19765       IF Do_plot$="TRUE" THEN
19770            GRAPHICS OFF
19775            ALPHA ON      ! So the user knows what to do next, i.e., CONTINUE
19780       END IF
19785    RETURN
19790 ! -------------------------end-of-sub-routine----------------------------
19795 !
19800 ! ************************************************************************
19805 Graph_mag_label: !INITIALIZE ALL MAIN VARIABLES USE FOR DRAWING MAGN.    *
19810 ! ************************************************************************
19815 !                                                                        *
19820 ! MODULE NAME : GRAPH_MAG_LABEL                    DATE: 11 SEPTEMBER 1985 *
19825 !                                                                        *
19830 ! PURPOSE: To set all the variable used by the graph subroutine - for    *
19835 !          the normalize magnitude of the beam pattern                   *
```

```
19840 !                                                                      *
19845 ! INPUT  :  SEE BELOW                                                  *
19850 ! OUTPUT :  ALL THE ONE BELOW                                          *
19855 ! CALLS  : NIL                                                         *
19860 ! SIDE EFFECTS : NIL                                                   *
19865 ! REMARKS : A Scaling factor Sf can be used for al variables           *
19870 ! ************************************************************************
19875   Left=-180                !LEFT origin value of X
19880   Right=180                !RIGHT origin value of X
19885   Bottom=0                 !BOTTOM origin value of Y
19890   Top=1                    !TOP origin value of Y
19895   !
19900   Xsp=10                   !MINOR TICKS spacing for X
19905   Ysp=.1                   !MINOR TICKS spacing for Y
19910   Xmaj=6                   !MAJOR TICKS spacing, every 5 minor ticks
19915   Ymaj=2                   !MAJOR TICKS spacing, every 5 minor ticks
19920   Majsz=2                  !size or length of major ticks
19925   !
19930   Xgrid=4.5*Xsp              !grid every Xgrid on X
19935   Ygrid=2*Ysp              !grid every Ygrid on Y
19940   !
19945   Ystepsz=ABS(Ysp)     !step to label Y axis, same units as top & bottom
19950   Xstepsz=Xsp          !step to label X axis, same units as left & right
19955   RETURN
19960   !
19965   ! N.B. AXES XTICK,YTICK,XLOCY,YLOCX,XMAJOR,YMAJOR,SIZE
19970   !      XLOCY, YLOCX SPECIFY THE LOCATION OF THE INTERSECTION OF
19975   !      THE AXES. THE SAME APPLY FOR GRID
19980   ! -------------------------end-of-sub-routine-------------------------
19985   !
19990   ! ************************************************************************
19995 Plot_db_bp:!        PLOT MAGNITUDE IN dB OF THE BEAM PATTERN            *
20000   ! ************************************************************************
20005   !                                                                      *
20010   ! MODULE NAME: PLOT_DB_BP                        DATE: 12 SEPTEMBER 1985  *
20015   !                                                                      *
20020   ! PURPOSE : Plot the graph of the magnitude of the beam pattern versus  *
20025   !           Horizontal angle (degrees)                                 *
20030   !                                                                      *
20035   ! INPUT  :  Used_mra, Graph_scale, Do_plot$, Theory_db_bp,            *
20040   !           Actual_db_bp, Line_type_theo, Line_type_actl.              *
20045   ! OUTPUT : Graph of Psi VERSUS Mag_db_bp                               *
20050   ! CALLS  : Gvsblabel, Insert_bot_line                                  *
20055   ! SIDE EFFECTS : NIL                                                   *
20060   ! REMARKS : 1. PEN 1 : EVERYTHING EXCEPT TITLE AND CURVES             *
20065   !              PEN 2 : FOR TITLE                                       *
20070   !              PEN 3 : FOR CURVES                                      *
20075   !                                                                      *
20080   ! ************************************************************************
20085   !
20090     GOSUB Graph_db_label                  ! Labels plot
20095     OUTPUT 2;Clear$;                      ! Clear Screen
20100     ALPHA OFF
20105     GCLEAR                                ! Clear Graphics Screen
20110     GRAPHICS ON
20115     GINIT                                 ! Initialize default graphics
20120     IF Do_plot$="TRUE " THEN
20125         PLOTTER IS 705,"HPGL"
20130     END IF
20135     DEG                                   ! All angles are in degrees
```

```
20140     !
20145     ! ------------------------------------------------------------------
20150     !
20155     PEN 1
20160     VIEWPORT 0,INT(Graph_scale*133),0,INT(Graph_scale*100)
20165     FRAME                                  ! Main frame is X = 0 TO 133
20170                                            !         AND    Y = 0 TO 100
20175     !     LABEL THE GRAPH
20180     !
20185     LORG 6                                 ! or any positive pen
20190     MOVE 67,98
20195     CSIZE INT(Graph_scale*6)
20200     PEN 2
20205     LABEL "HORIZONTAL BEAM PATTERN IN dB"      ! MAIN TITLE
20210     MOVE 67,91
20215     CSIZE INT(Graph_scale*4)
20220     PEN 1
20225     LABEL "MRA : ";Used_mra;" degrees"     ! SUBTITLE
20230     MOVE 4,50
20235     LDIR 90                                ! rotate 90 Degrees
20240     LABEL "magnitude in dB"
20245     LDIR 0                                 ! back to 0 Degree
20250     !
20255     ! ------------------------------------------------------------------
20260     !
20265     GOSUB Insert_bot_line   ! Insert line of info at the bottom of graph
20270     !
20275     ! ------------------------------------------------------------------
20280     !
20285     GOSUB Insert_legend     ! Insert the legend for this graph
20290     !
20295     ! ------------------------------------------------------------------
20300     !
20305     !           LABEL THE X AXIS
20310     LORG 5
20315     MOVE 67,7
20320         LABEL "HORIZONTAL ANGLE (Degrees)"
20325     !
20330     ! ------------------------------------------------------------------
20335     !
20340     ! CREATE THE FRAME WHICH WILL CONTAINS THE ACTUAL GRAPH
20345     !
20350     VIEWPORT INT(Graph_scale*15),INT(Graph_scale*115),INT(Graph_scale*12),
INT(Graph_scale*87)
20355     WINDOW Left,Right,Bottom,Top
20360     AXES Xsp,Ysp,Left,Right,Xmaj,Ymaj,Majsz     ! draw axes LEFT & RIGHT
20365     AXES Xsp,Ysp,Right,Top,Xmaj,Ymaj,Majsz      ! draw axes TOP & RIGHT
20370     AXES Xsp,Ysp,0,Bottom,Xmaj,Ymaj,Majsz       ! draw axes LEFT & BOTTOM
20375     !
20380     LINE TYPE 4                            ! dotted line
20385     !   No grid line are outputed if a plot is done
20390     IF Do_plot$="FALSE" THEN
20395          GRID Xgrid,Ygrid
20400     END IF
20405     LINE TYPE 1                            ! continuous line
20410     FRAME
20415     LORG 8
20420     CSIZE INT(Graph_scale*3)
20425     CLIP OFF
20430     !
```

```
20435     FOR L=Bottom TO Top STEP Ystepsz       ! indicate Y steps
20440          MOVE Left,L
20445          LABEL L
20450     NEXT L  !
20455     !
20460     ! ---------------------------------------------------------------------
20465     !
20470     !   OUTPUT THE X DIVISIONS FOR THE HORIZONTAL ANGLE
20475     !
20480     LORG 6
20485     FOR L=Left TO Right STEP (Xstepsz*4)      ! indicate X steps
20490          MOVE L,Bottom
20495          LABEL L
20500     NEXT L
20505     !
20510     ! ---------------------------------------------------------------------
20515     !
20520     !    DRAW THE CURVE
20525     !
20530     CSIZE INT(Graph_scale*5)
20535     CLIP ON                          ! Keep the curves inside the window
20540     MOVE Psi(-180),Theory_db_bp(-180)   ! IMPORTANT, RETURN TO FIRST POINT
20545     !
20550     !*****************************************************
20555     ! THEORETICAL CURVE
20560     LINE TYPE Line_type_theo
20565     PEN 3
20570     FOR J=-180 TO 180 STEP 1
20575          DRAW Psi(J),Theory_db_bp(J)
20580     NEXT J
20585     !
20590     !*****************************************************
20595     ! ACTUAL OR DEGRADED CURVE
20600     LINE TYPE Line_type_actl
20605     PEN 3
20610     MOVE Psi(-180),Actual_db_bp(-180)
20615     FOR K=-180 TO 180 STEP 1
20620          DRAW Psi(K),Actual_db_bp(K)
20625     NEXT K
20630     !
20635     LDIR 0
20640     PEN 0                                ! PEN UP
20645     IF Do_plot$="TRUE" THEN
20650          GRAPHICS OFF
20655          ALPHA ON      ! So the user knows what to do next, i.e., CONTINUE
20660     END IF
20665   RETURN
20670 ! -------------------------end-of-sub-routine---------------------------
20675 !
20680 !
20685 ! ***************************************************************************
20690 Graph_db_label: !INITIALIZE ALL MAIN VARIABLES USE FOR DRAWING MAGN.      *
20695 ! ***************************************************************************
20700 !                                                                          *
20705 ! MODULE NAME : GRAPH_DB_LABEL                    DATE: 12 SEPTEMBER 1985  *
20710 !                                                                          *
20715 ! PURPOSE: To set all the variable used by the graph subroutine - for     *
20720 !          the normalize magnitude of the beam pattern                     *
20725 !                                                                          *
20730 ! INPUT  :  SEE BELOW                                                      *
```

```
20735 ! OUTPUT :  ALL THE ONE BELOW                                          *
20740 ! CALLS  : NIL                                                        *
20745 ! SIDE EFFECTS : NIL                                                  *
20750 ! REMARKS : A Scaling factor Sf can be used for al variables          *
20755 ! ***************************************************************************
20760   Left=-180                 !LEFT origin value of X
20765   Right=180                 !RIGHT origin value of X
20770   Bottom=Minimum_nn_db      ! For graph with magnitude in dB
20775   Top=0
20780   !
20785   Xsp=10                    !MINOR TICKS spacing for X
20790   Ysp=Minimum_nn_db/10
20795   Xmaj=6                    !MAJOR TICKS spacing, every 5 minor ticks
20800   Ymaj=5                    !MAJOR TICKS spacing, every 5 minor ticks
20805   Majsz=2                   !size or length of major ticks
20810   !
20815   Xgrid=4.5*Xsp               !grid every Xgrid on X
20820   Ygrid=2*Ysp                 !grid every Ygrid on Y
20825   !
20830   Ystepsz=ABS(Ysp)      !step to label Y axis, same units as top & bottom
20835   Xstepsz=Xsp           !step to label X axis, same units as left & right
20840   RETURN
20845   !
20850   ! N.B. AXES XTICK,YTICK,XLOCY,YLOCX,XMAJOR,YMAJOR,SIZE
20855   !      XLOCY, YLOCX SPECIFY THE LOCATION OF THE INTERSECTION OF
20860   !      THE AXES. THE SAME APPLY FOR GRID
20865 ! ------------------------end-of-sub-routine--------------------------
20870 !
20875  ! ***************************************************************************
20880 Plot_po_bp: ! PLOT THE POLAR REPRSENTATION OF THE BEAM PATTERN          *
20885  ! ***************************************************************************
20890  !                                                                      *
20895  ! MODULE NAME: PLOT_PO_DB                      DATE: 12 SEPTEMBER 1985  *
20900  !                                                                      *
20905  ! PURPOSE : Plot the POLAR graph of the magnitude of the beam pattern  *
20910  !           in dB as function of the horizontal angle (psi).           *
20915  !                                                                      *
20920  ! INPUT  : Used_mra, Minimum_nn_db, Graph_scale, Theory_db_xaxis,      *
20925  !          Theory_db_yaxis, Actual_db_xaxis, Actual_db_yaxis,          *
20930  !          From_angle_1, From_angle_2, To_angle_1, To_angle_2,         *
20935  !          From_angle_1_r, From_angle_2_r, To_angle_1_r, To_angle_2_r  *
20940  ! OUTPUT : Polar plot of the horizontal beam pattern                   *
20945  ! CALLS  : Graph_po_label, Insert_bot_line                            *
20950  ! SIDE EFFECTS : NIL                                                  *
20955  ! REMARKS : 1. PEN 1 : ALL FEATURES EXCEPT TITLE, AND DRAWING        *
20960  !           2. PEN 2 : FOR TITLE                                      *
20965  !           3. PEN 3 : FOR DRAWING THE BEAM PATTERN CURVES            *
20970  !                                                                     *
20975  ! ***************************************************************************
20980  !
20985     ! local database
20990     INTEGER Oblevel                  ! Used to plot the dB level
20995     !
21000     ! ---------------------------------------------------------------------
21005     GOSUB Graph_po_label
21010     OUTPUT 2;Clear$;                 ! CLEAR THE SCREEN
21015     PRINT USING "@"                  ! GO TO THE END OF THE PAGE
21020     ALPHA OFF
21025     GCLEAR                           ! SAME AS OUTPUT 2;CLEAR$;
21030     GRAPHICS ON
```

```
21035     GINIT
21040     IF Do_plot$="TRUE " THEN
21045         PLOTTER IS 705,"HPGL"
21050     END IF
21055     DEG                              ! USE ANGLES IN DEGREES
21060     !
21065     ! ---------------------------------------------------------------
21070     !
21075     PEN 1
21080     IF Graph_scale>.90 THEN
21085          VIEWPORT 0,133,0,100        ! Maximum size
21090     ELSE
21095          VIEWPORT 0,100,0,82         ! Reduce size OR 0,100,0,82
21100     END IF
21105     FRAME
21110     !
21115     !        LABEL THE GRAPH
21120     PEN 2
21125     LORG 6
21130     MOVE 67,99
21135     IF Graph_scale>.90 THEN
21140          CSIZE 5
21145     ELSE
21150          CSIZE 4
21155     END IF
21160     LABEL "MAG. OF HORIZONTAL BEAM PATTERN"          ! MAIN TITLE
21165     IF Graph_scale>.9 THEN
21170          CSIZE 3
21175     ELSE
21180          CSIZE 2
21185     END IF
21190     PEN 1
21195     LABEL "MRA :";Used_mra;" degrees"
21200     !
21205     ! ---------------------------------------------------------------
21210     !
21215     GOSUB Insert_bot_line   ! Insert the required details at the bottom
21220     !
21225     ! ---------------------------------------------------------------
21230     !
21235     GOSUB Insert_legend     ! Insert the legend of this graph
21240     !
21245     ! ---------------------------------------------------------------
21250     !
21255     !    DEFINE THE FRAME WHICH WILL CONTAIN THE ACTUAL GRAPH
21260     !
21265       ! For the following it is important to have
21270       ! (Right-Left = 2(Top-Bottom). e.g. 123-9 = 2(72-15)
21275       !
21280     IF Graph_scale>.90 THEN
21285          VIEWPORT 25,106,7,88       ! Largest reasonable size
21290     ELSE
21295          VIEWPORT 20,80,8,68        ! Reduce size
21300     END IF
21305     !
21310     WINDOW Left,Right,Bottom,Top
21315     FRAME
21320     AXES Xsp,Ysp,(Right+Left)/2,(Bottom+Top)/2,Xmaj,Ymaj,Majsz   ! BOTTOM
& CENTER
21325     LINE TYPE 1
```

```
21330     !
21335     ! ---------------------------------------------------------------------
21340     !
21345     !   DRAW THE HALF-CIRCLES
21350     !
21355      FOR J=0 TO -(Minimum_nn_db) STEP 10
21360           Arc(((Right+Left)/2),((Bottom+Top)/2),(J),0,360)
21365      NEXT J
21370     !
21375     ! ---------------------------------------------------------------------
21380     !
21385     !   DRAW THE ANGLE LINES
21390     !
21395     CSIZE 3
21400     LINE TYPE 1
21405     FOR Angle=-180 TO 180 STEP 15
21410          MOVE (Right+Left)/2,(Bottom+Top)/2
21415         !PIVOT 90-Angle     ! START FROM QUADRANT 3 TO 2,1, THEN 4.
21420          PIVOT Angle
21425          IDRAW 55,0
21430          !
21435          !  INSERT ANGLE LABEL AT THE CORRECT LOCATION
21440          !    AROUND THE GRAPH
21445          !
21450            PIVOT 0              ! RESTORE NORMAL ORIENTATION OF THE GRAPH
21455            CLIP OFF
21460            IF (Angle<45) AND (Angle>-45) THEN
21465                IF (Angle<=0) THEN
21470                  IF (Angle<+.001) AND (Angle>-.001) THEN
21475                     Y_position=0
21480                  ELSE
21485                     Y_position=-(Right)*TAN(-Angle)
21490                  END IF
21495                ELSE
21500                   Y_position=(Right)*TAN(Angle)  ! For BOTTOM
21505                END IF   ! (ANGLE<=0)
21510                LORG 2             ! LABEL AT RIGHT HAND SIDE OF +
21515                MOVE Right,Y_position
21520                LABEL 90-Angle
21525            ELSE
21530                IF (Angle>=45) AND (Angle<=135) THEN
21535                   LORG 4          ! LABEL ABOVE CENTER +
21540                   IF Angle<(90-.001) THEN
21545                        X_position=Top*TAN(90-Angle)
21550                        MOVE X_position,Top
21555                        LABEL 90-Angle
21560                   ELSE
21565                        IF (Angle>(90-.001)) AND (Angle<(90+.001)) THEN !
i.e.ZERO degrees
21570                            X_position=0
21575                        ELSE
21580                            X_position=(-Top*TAN(Angle-90))
21585                        END IF
21590                        MOVE X_position,Top
21595                        LABEL 90-Angle
21600                   END IF          ! ANGLE <= 90
21605                ELSE
21610                   IF (Angle<-135) OR (Angle>135) THEN
21615                       LORG 8           ! LABEL LEFT HAND SIDE OF +
21620                       IF (Angle<=-135) THEN
```

```
21625                         Y_position=(Left)*TAN(180+Angle)   ! For TOP
21630                         MOVE Left,Y_position
21635                         LABEL -270-Angle
21640                       ELSE
21645                          IF (Angle<180) THEN
21650                             Y_position=(Top)*TAN(180-Angle) ! For BOTTOM
21655                             MOVE Left,Y_position
21660                             LABEL 90-Angle
21665                          END IF ! ANGLE< 180
21670                       END IF    ! (ANGLE<=-135)
21675                  ELSE ! From -135 TO +135
21680                       LORG 6
21685                       IF Angle<(-90-.001) THEN
21690                            X_position=Bottom*TAN(-(Angle+90))
21695                             MOVE X_position,Bottom
21700                             LABEL -270-Angle
21705                       ELSE
21710                            IF (Angle>(-90-.001)) AND (Angle<(-90+.001))
THEN
21715                               X_position=0
21720                            ELSE
21725                               X_position=(Top*TAN(90+Angle))
21730                            END IF
21735                            MOVE X_position,Bottom
21740                            LABEL 90-Angle
21745                       END IF          ! ANGLE <= 90
21750                  !
21755                  END IF         !  (ANGLE>135) AND (ANGLE<225)
21760              END IF     ! ANGLE >= 45 AND ANGLE <= 135
21765          END IF   !  Angle < 45
21770          CLIP ON
21775          !
21780          NEXT Angle
21785     !
21790     ! ---------------------------------------------------------------------
21795     !
21800     ! LABEL THE CENTRAL AXIS
21805     !
21810     LORG 4
21815     FOR Dblevel=Top-10 TO (Top+Bottom)+.1 STEP -10
21820         MOVE (Left+Right),Dblevel
21825         LABEL Minimum_nn_db+Dblevel
21830     NEXT Dblevel
21835     !
21840     FOR Dblevel=Bottom+10 TO (Top+Bottom)-.1 STEP 10
21845         MOVE (Left+Right),Dblevel
21850         LABEL -Dblevel+Minimum_nn_db
21855     NEXT Dblevel
21860     !
21865     ! ADD INDICATION OF 0 dB
21870     !
21875     MOVE (Left+Right),Bottom
21880     LABEL " 0 DB"
21885     !
21890     ! ---------------------------------------------------------------------
21895     !
21900     !       DRAW THE BEAM PATTERN
21905     !
21910     PEN 3
21915     CALL Angle_defn((Used_mra),From_angle_1_r,To_angle_1_r,From_angle_2_r,
```

```
To_angle_2_r)
21920     From_angle_1=INT(From_angle_1_r)
21925     To_angle_1=INT(To_angle_1_r)
21930     From_angle_2=INT(From_angle_2_r)
21935     To_angle_2=INT(To_angle_2_r)
21940     !*************************************************
21945     ! THEORETICAL CURVE
21950     LINE TYPE Line_type_theo
21955     MOVE Theory_db_xaxis(From_angle_1),Theory_db_yaxis(From_angle_1) ! IMP
ORTANT, RETURN TO THE 1ST PT
21960     FOR J=From_angle_1 TO To_angle_1 STEP 1
21965           DRAW Theory_db_xaxis(J),Theory_db_yaxis(J)
21970     NEXT J
21975     IF (From_angle_2=0) AND (To_angle_2=0) THEN
21980         MOVE Theory_db_xaxis(From_angle_2),Theory_db_yaxis(From_angle_2)
21985         FOR J=From_angle_2 TO To_angle_2 STEP 1
21990               DRAW Theory_db_xaxis(J),Theory_db_yaxis(J)
21995         NEXT J
22000     END IF ! NO second angle
22005     !
22010     !*************************************************
22015     ! ACTUAL OR DEGRADED CURVE
22020     LINE TYPE Line_type_actl
22025     MOVE Actual_db_xaxis(From_angle_1),Actual_db_yaxis(From_angle_1)
22030     FOR K=From_angle_1 TO To_angle_1 STEP 1
22035           DRAW Actual_db_xaxis(K),Actual_db_yaxis(K)
22040     NEXT K
22045     IF (From_angle_2=0) AND (To_angle_2=0) THEN
22050           MOVE Actual_db_xaxis(From_angle_2),Actual_db_yaxis(From_angle_2)
22055           FOR K=From_angle_2 TO To_angle_2 STEP 1
22060               DRAW Actual_db_xaxis(K),Actual_db_yaxis(K)
22065           NEXT K
22070     END IF ! NO second angle
22075     PEN 0
22080     IF Do_plot$="TRUE " THEN
22085           GRAPHICS OFF
22090           ALPHA ON          ! So user knows what to do next, i.e., CONTINUE
22095     END IF
22100  RETURN ! Plot_po_bp
22105 !
22110 ! ************************************************************************
22115 !
22120 ! ************************************************************************
22125 Graph_po_label:  ! INITIALIZE ALL MAIN VARIABLES USE FOR DRAWING POLAR.  *
22130 ! ************************************************************************
22135 !                                                                        *
22140 ! MODULE NAME : GRAPH_PO_LABEL                    DATE: 12 SEPTEMBER 1985  *
22145 !                                                                        *
22150 ! PURPOSE: To set all the variable used by the graph subroutine - for    *
22155 !          the polar plot of the Beam Pattern expressed in dB.           *
22160 !                                                                        *
22165 ! INPUT  :  SEE BELOW                                                   *
22170 ! OUTPUT :  ALL THE ONE BELOW                                           *
22175 ! CALLS  : NIL                                                         *
22180 ! SIDE EFFECTS : NIL                                                   *
22185 ! REMARKS : A Scaling factor Sf can be used for al variables            *
22190 ! ************************************************************************
22195    Left=Minimum_nn_db      ! LEFT origin value of X
22200    Right=-Left             ! RIGHT origin value of X
22205    Top=-Minimum_nn_db
```

```
22210    Bottom=Minimum_nn_db
22215    Xsp=(Right-Left)/16        ! MINOR ticks spacing for X
22220    Ysp=-5                     ! MINOR ticks spacing for Y
22225    Xmaj=5                     !MAJOR TICKS spacing, every 5 minor ticks
22230    Ymaj=5                     !MAJOR TICKS spacing, every 5 minor ticks
22235    Majsz=2                    !size or length of major ticks
22240    !
22245    Xgrid=(2*Xsp)              !grid every Xgrid on X
22250    Ygrid=2*Ysp                !grid every Ygrid on Y
22255    !
22260    Ystepsz=ABS(Ysp)       !step to label Y axis, same units as top & bottom
22265    Xstepsz=Xsp            !step to label X axis, same units as left & right
22270   RETURN  ! GRAPH_PO_LABEL
22275   !
22280   ! N.B. AXES XTICK,YTICK,XLOCY,YLOCX,XMAJOR,YMAJOR,SIZE
22285   !      XLOCY, YLOCX SPECIFY THE LOCATION OF THE INTERSECTION OF
22290   !      THE AXES. THE SAME APPLY FOR GRID
22295 ! ------------------------end-of-sub-routine--------------------------
22300 !
22305 ! ****************************************************************************
22310 Terminate:!    TERMINATE ELEGANTLY THE PROGRAM                                *
22315 ! ****************************************************************************
22320 !                                                                            *
22325 ! MODULE NAME: TERMINATE                          DATE: 26 NOVEMBER 1985    *
22330 !                                                                            *
22335 ! PROGRAMMER: Sylvain Fleurant                    DATE: 26 NOVEMBER 1985    *
22340 !                                                                            *
22345 ! PURPOSE: To terminate elegantly the program by clearing the screen        *
22350 !          and informing the user about how to re-run the program.           *
22355 !                                                                            *
22360 ! INPUT: NIL                                                                *
22365 ! OUTPUT: Message on the screen                                             *
22370 ! SIDE EFFECTS: Clear the screen and print a message on the CRT.            *
22375 ! REMARKS: NIL                                                              *
22380 ! ****************************************************************************
22385      GRAPHICS OFF
22390      OUTPUT 2;Clear$;
22395      ALPHA ON
22400      PRINT "PROGRAM TERMINATION"
22405      PRINT USING "//"
22410      PRINT "Press RUN to re-start the program to compute new beam pattern"
22415      PRINT " "
22420  END ! OF SUB_ARRAY
22425 ! ****************************************************************************
22430 SUB Fft4(Sign,Xr(*),Xi(*),INTEGER N_power,INTEGER Nmax) !                    *
22435 ! ****************************************************************************
22440 !                                                                            *
22445 ! SUBPROGRAM NAME: FFT4                              DATE 24 JAN 1985      *
22450 !                                                                            *
22455 ! ORIGINAL PROGRAMMER: Prof J. Walters                                      *
22460 !                                                                            *
22465 ! REV 1.1: CHANGES                                                         *
22470 !          1.  Reverse the type of calculation from indirect to direct     *
22475 !              and vice and versa. Add basic documentation.                  *
22480 !                                                                            *
22485 ! PURPOSE: To compute the direct or indirect Fast Fourier Transform        *
22490 !          of a complex array (real and imaginary).                          *
22495 !                                                                            *
22500 ! INPUT  : XR(*), XI(*), SIGN, NMAX, & N_POWER                             *
22505 ! OUTPUT : XR(*), XI(*)                                                     *
```

```
22510 ! CALLS  : NIL                                                          *
22515 ! SIDE EFFECTS : NIL                                                    *
22520 ! REMARKS:                                                              *
22525 !          SIGN = -1. Direct Transform                                  *
22530 !          SIGN = +1. Indirect Transform                                *
22535 !          Xr   = The real array                                        *
22540 !          Xi   = The imaginary array                                   *
22545 !          N_power = The power of 2 for the transform                   *
22550 !          Nmax = The size of the Xr and Xi arrays (2^N_power)          *
22555 !          Inserted for future version if needed.                       *
22560 !                                                                       *
22565 ! ***********************************************************************
22570 !
22575 OPTION BASE 1                                  ! ALL ARRAYS START AT 1
22580 INTEGER I,Ii,Ij,J,Layer,Ll,Loc,Nn,Nw,Mm,Msk(1:13)
22585 Pts=Nmax
22590 Zz=2.*PI*Sign/Pts
22595 Msk(1)=Nmax DIV 2
22600 FOR I=2 TO N_power
22605   Msk(I)=Msk(I-1) DIV 2
22610 NEXT I
22615 Nn=Nmax
22620 Mm=2
22625 FOR Layer=1 TO N_power
22630   Nn=Nn DIV 2
22635   Nw=0
22640   FOR I=1 TO Mm STEP 2
22645     Ii=Nn*I
22650     W=Nw*Zz
22655     Cr=COS(W)
22660     Ci=SIN(W)
22665     FOR J=1 TO Nn
22670       Ii=Ii+1
22675       Ij=Ii-Nn
22680       Xar=Cr*Xr(Ii)-Ci*Xi(Ii)
22685       Xai=Cr*Xi(Ii)+Ci*Xr(Ii)
22690       Xr(Ii)=Xr(Ij)-Xar
22695       Xi(Ii)=Xi(Ij)-Xai
22700       Xr(Ij)=Xr(Ij)+Xar
22705       Xi(Ij)=Xi(Ij)+Xai
22710     NEXT J
22715     FOR Loc=2 TO N_power
22720       Ll=Nw-Msk(Loc)
22725       IF Ll<1 THEN GOTO Label1
22730       Nw=Ll
22735     NEXT Loc
22740 Label1: !
22745     IF Ll=0 THEN
22750       Nw=Msk(Loc+1)
22755     ELSE
22760       Nw=Msk(Loc)+Nw
22765     END IF
22770   NEXT I
22775   Mm=Mm*2
22780 NEXT Layer
22785   Nw=0
22790   FOR I=1 TO Nmax
22795     Nw1=Nw+1
22800     Xhr=Xr(Nw1)
22805     Xhi=Xi(Nw1)
```

```
22810     IF Nw1<I THEN GOTO Label3
22815     IF Nw1=I THEN GOTO Label2
22820       Xr(Nw1)=Xr(I)
22825       Xi(Nw1)=Xi(I)
22830 Label2:!
22835       Xr(I)=Xhr
22840       Xi(I)=Xhi
22845 Label3: !
22850     FOR Loc=1 TO N_power
22855       L1=Nw-Msk(Loc)
22860       IF L1<1 THEN GOTO Label4
22865       Nw=L1
22870     NEXT Loc
22875 Label4: !
22880     IF L1=0 THEN
22885       Nw=Msk(Loc+1)
22890     ELSE
22895       Nw=Msk(Loc)+Nw
22900     END IF
22905   NEXT I
22910   IF Sign<0. THEN
22915      FOR I=1 TO Nmax
22920        Xr(I)=Xr(I)/Pts
22925        Xi(I)=Xi(I)/Pts
22930      NEXT I
22935    END IF
22940  SUBEND    ! OF FFT
22945  ! ------------------------end-of-sub-program--------------------------
22950  !
22955  ! ****************************************************************************
22960 SUB Polar_form(Xr(*),Xi(*),Magn_x(*),Phase_x(*),INTEGER Nmax)
22965 ! ****************************************************************************
22970 !                                                                            *
22975 ! SUBPROGRAM NAME: POLAR_FORM                      DATE: 24 MAY 1985       *
22980 !                                                                            *
22985 ! PROGRAMMER: Sylvain Fleurant                                              *
22990 !                                                                            *
22995 ! PURPOSE: To convert a real and imaginary arrays to polar form, i.e.      *
23000 !          magnitude and phase.                                              *
23005 !                                                                            *
23010 ! INPUT  : XR(*), XI(*), MAGN_X(*), PHASE_X(*), NMAX                       *
23015 ! OUTPUT : MAGN_X(*), PHASE_X(*)                                           *
23020 ! CALLS  : NIL                                                             *
23025 ! SIDE EFFECTS : NIL                                                       *
23030 ! REMARKS: NIL                                                             *
23035 ! ****************************************************************************
23040 !
23045 ! DATABASE DEFINITION
23050  INTEGER Sign                    ! USED TO OBTAIN PROPER SIGN OF ANGLE
23055  INTEGER I                       ! USED FOR INDEXING PURPOSES
23060 !
23065  FOR I=-Nmax TO Nmax
23070       Magn_x(I)=SQR(Xr(I)*Xr(I)+Xi(I)*Xi(I))
23075       IF (ABS(Xr(I))<=1.0E-50) THEN
23080           IF (ABS(Xi(I))<=1.0E-50) THEN       ! i.e. both are 0
23085                Phase_x(I)=0.
23090           ELSE
23095                Sign=Xi(I)/(ABS(Xi(I)))        ! + OR - infinity
23100                Phase_x(I)=ATN(Sign*1.0E+50)   ! ATN = ARCTAN
23105           END IF
```

```
23110       ELSE
23115           IF (ABS(Xi(I))<=1.E-50) AND (Xr(I)<0) THEN
23120               Phase_x(I)=PI
23125           ELSE
23130               Phase_x(I)=ATN(Xi(I)/Xr(I))     ! ARCTAN
23135           END IF
23140       END IF
23145       Phase_x(I)=Phase_x(I)*180/PI            ! CONVERT TO DEGREE
23150  NEXT I
23155 SUBEND
23160 ! ---------------------------end-of-sub-program--------------------------
23165 !
23170 ! ****************************************************************************
23175 SUB Rect_form(Xr(*),Xi(*),Magn_x(*),Phase_x(*),INTEGER Nmax)
23180 ! ****************************************************************************
23185 ! SUBPROGRAM NAME: RECTANGULAR_FORM                DATE: 25 MAY 1985       *
23190 !                                                                           *
23195 ! PROGRAMMER: Sylvain Fleurant                                             *
23200 !                                                                           *
23205 ! PURPOSE: To convert from polar form i.e. magnitude and phase the         *
23210 !          elements of the array x to rectangular form i.e. real           *
23215 !          and imaginary components.                                       *
23220 !                                                                           *
23225 ! INPUT  : XR(*), XI(*), MAGN_X(*), PHASE_X(*), NMAX                       *
23230 ! OUTPUT : XR(*),XI(*)                                                     *
23235 ! CALLS  : NIL                                                            *
23240 ! SIDE EFFECTS : NIL                                                      *
23245 ! REMARKS: The phase must be in radians to produce correct values         *
23250 ! ****************************************************************************
23255 !
23260 ! DATABASE DEFINITION
23265  INTEGER I                          ! USED FOR INDEXING PURPOSES
23270 !
23275  RAD                               ! Angles are in radians
23280  FOR I=-Nmax TO Nmax
23285        Xr(I)=Magn_x(I)*COS(Phase_x(I))
23290        Xi(I)=Magn_x(I)*SIN(Phase_x(I))
23295  NEXT I
23300 !
23305 SUBEND
23310 ! ---------------------------end-of-sub-program--------------------------
23315 !
23320 ! ****************************************************************************
23325 DEF FNIs_string_pnum$(Data_line$) ! IS STRING A POSITIVE NUMBER
23330 ! ****************************************************************************
23335 !                                                                           *
23340 ! SUBFUNCTION NAME: IS_STRING_PNUM                DATE: 27 JULY 1985      *
23345 !                                                                           *
23350 ! PROGRAMMER: Sylvain Fleurant                                             *
23355 !                                                                           *
23360 ! PURPOSE: To evaluate if the data_line (a long string) is a positive     *
23365 !          number i.e. not blank or formed of characters. The number      *
23370 !          can be positive or negative, and precedeted by blanks. Only    *
23375 !          the first number is checked, any following alphanumeric        *
23380 !          string is discarded.                                           *
23385 !                                                                         *
23390 ! INPUT  : Data_line$ (assumed to be up to 160 long)                      *
23395 !          (Preferably passed by values)                                  *
23400 ! OUTPUT : Ok_input$ (TRUE  or FALSE)                                     *
23405 ! CALLS  : NIL                                                           *
```

```
23410 ! SIDE EFFECTS : NIL                                                    *
23415 ! REMARKS: 1. TRUE  means the string is a positive number               *
23420 !                   or a number can be extracted once); and             *
23425 !             FALSE means the string is not a number (i.e. all blanks   *
23430 !                   or nonnumeric).                                     *
23435 !                                                                       *
23440 !          2. A string of 5 char. is returned.                          *
23445 ! ***************************************************************************
23450 !
23455 ! LOCAL DATABASE DEFINITION
23460 !
23465   REAL Temporary_real           ! Temporary REAL variable
23470   ALLOCATE Ok_input$[5]         ! The result of the test - TRUE or FALSE
23475 !
23480   Ok_input$="FALSE"             ! Default value
23485   IF NOT LEN(Data_line$) OR (NOT LEN(TRIM$(Data_line$))) THEN !It is empty
23490         Ok_input$="FALSE"
23495   ELSE
23500         !
23505         Data_line$=TRIM$(Data_line$) !Remove all leading & trailing blanks
23510         ! Check the first character of the string
23515         IF NOT (NUM(Data_line$[1,1])=NUM("-")) THEN
23520               Temporary_real=NUM(Data_line$[1,1])
23525         ELSE
23530               Temporary_real=NUM(Data_line$[2,2]) ! Skip the "-"
23535         END IF    ! of not "-"
23540         !
23545         ! Check if the corresponding ASCII of the first char
23550         ! is > then 9 or < then 0
23555         !
23560         IF (Temporary_real>NUM("9")) OR (Temporary_real<NUM("0")) THEN
23565               ! This is a character not a number
23570               Ok_input$="FALSE"
23575         ELSE     ! This is a numeric string
23580            ENTER Data_line$;Temporary_real
23585            IF Temporary_real<0 THEN     ! This is a negative number
23590              Ok_input$="FALSE"
23595            ELSE
23600              Ok_input$="TRUE "
23605            END IF    ! of temporary_real < 0
23610         END IF     ! of numeric
23615    END IF ! not len
23620    RETURN Ok_input$
23625 !
23630 FNEND ! of Is_string_pnum
23635 !
23640 ! --------------------End-of-Sub-Function--------------------------------
23645 !
23650 ! ***************************************************************************
23655 Arc: SUB Arc(X,Y,OPTIONAL Radius_,Start_,End_,Intervals_,Penup_,Aspect_)
23660 ! ***************************************************************************
23665 !                                                                       *
23670 ! SUBPROGRAM NAME: ARC                                 DATE: 18 JULY 1985    *
23675 !                                                                       *
23680 ! PROGRAMMER : HP Packages                                             *
23685 !                                                                       *
23690 ! PURPOSE: This subroutine draws an arc of a circle with the center at  *
23695 !          X, Y and a radius "Radius_". See below for details.          *
23700 !                                                                       *
23705 ! INPUT  : X, Y, OPTIONAL Radius_, Start_, End_, Penup_, Aspect_        *
```

```
23710 ! OUTPUT : DRAWING ARC                                                  *
23715 ! SIDE EFFECTS: NIL                                                     *
23720 ! REMARKS:   DETAIL USAGES:                                             *
23725 !                                                                       *
23730 !       1. The arc starts at a position of "Start" degrees and ends at  *
23735 !          "End" degrees and has a total of "Interval"individual line   *
23740 !          segments. The greater "Intervals" is, the rounder the arc will*
23745 !          look, but also the longer the routine will take to finish.   *
23750 !                                                                       *
23755 !       2. If "Penup" is non-zero, the pen will be picked up before the *
23760 !          arc is started.  If not, it will be left down (assuming it was*
23765 !          down before).                                                *
23770 !                                                                       *
23775 !       3. Oftentimes, you want to draw a straight line to the arc you  *
23780 !          are starting to draw. If "Radius" is positive, the arc will  *
23785 !          proceed counterclokwise; if negative, clockwise.             *
23790 !                                                                       *
23795 ! ***************************************************************************
23800 !
23805    ON 9-NPAR GOTO Aspect_lbl,Penup_lbl,Interval_lbl,End_lbl,Start_lbl,Radi
us_lbl,Standard_lbl              ! ON <maxparms>+1-NPAR
23810    !
23815 Aspect_lbl:    Aspect=Aspect_
23820 Penup_lbl:     Penup=Penup_
23825 Interval_lbl:  Intervals=Intervals_
23830 End_lbl:       End=End_
23835 Start_lbl:     Start=Start_
23840 Radius_lbl:    Radius=Radius_
23845 Standard_lbl:  ON NPAR-1 GOTO Radius_lbl2,Start_lbl2,End_lbl2,Intervals_lb
l2,Penup_lbl2,Aspect_lbl2,Degree_lbl2       ! NAPR+1-<req , parms>
23850     !
23855     ! Define the default values according to options
23860     !
23865 Radius_lbl2:   Radius=1.
23870 Start_lbl2:    Start=0.
23875 End_lbl2:      End=360.
23880 Intervals_lbl2:Intervals=INT((End-Start)/5.)
23885 Penup_lbl2:    Penup=1
23890 Aspect_lbl2:   Aspect=1.
23895 Degree_lbl2:   DEG
23900     !
23905     IF Penup THEN PENUP
23910     IF (Radius>0.) AND (End<=Start) THEN End=End+360
23915     IF (Radius<0.) AND (End>=Start) THEN End=End-360
23920     Step=(End-Start)/Intervals
23925     Radius=ABS(Radius)
23930     FOR I=Start TO End STEP Step
23935           PLOT X+Radius*Aspect*COS(I),Y+Radius*SIN(I)
23940     NEXT I
23945  SUBEND      ! End of arc
23950 ! -------------------------end-of-sub-program--------------------------
23955 !
23960 ! ***************************************************************************
23965 DEF FNIs_string_num$(Data_line$) ! IS STRING A NUMBER  ? (+ or -)        *
23970 ! ***************************************************************************
23975 ! SUBFUNCTION NAME: IS_STRING_NUM              DATE: 25 SEPTEMBER 1985    *
23980 !                                                                       *
23985 ! PROGRAMMER: Sylvain Fleurant                                         *
23990 !                                                                       *
23995 ! PURPOSE: To evaluate if the data_line (a long string) is a            *
```

```
24000 !          number i.e. not blank or formed of characters. The number     *
24005 !          can be positive or negative, and precedeted by blanks. Only   *
24010 !          the first number is checked, any following alphanumeric       *
24015 !          string is discarded.                                          *
24020 !                                                                        *
24025 ! INPUT  : Data_line$ (assumed to be up to 160 long)                     *
24030 !           (Preferably passed by values)                                *
24035 ! OUTPUT : Ok_input$ (TRUE  or FALSE)                                    *
24040 ! CALLS  : NIL                                                           *
24045 ! SIDE EFFECTS : NIL                                                     *
24050 ! REMARKS: 1. TRUE  means the string is a number postive or negative     *
24055 !                 (or a number can be extracted once); and               *
24060 !             FALSE means the string is not a number (i.e. all blanks    *
24065 !                   or nonnumeric).                                      *
24070 !                                                                        *
24075 !          2. A string of 5 char. is returned.                           *
24080 ! ***************************************************************************
24085 !
24090 ! LOCAL DATABASE DEFINITION
24095 !
24100   INTEGER Temporary_real        ! Temporary REAL variable
24105   ALLOCATE Ok_input$[5]         ! The result of the test - TRUE or FALSE
24110 !
24115   Ok_input$="FALSE"             ! Default value
24120   IF NOT LEN(Data_line$) OR (NOT LEN(TRIM$(Data_line$))) THEN !It is empty
24125         Ok_input$="FALSE"
24130   ELSE
24135         !
24140         Data_line$=TRIM$(Data_line$) !Remove all leading & trailing blanks
24145         ! Check the first character of the string
24150         IF NOT (NUM(Data_line$[1,1])=NUM("-")) THEN
24155               Temporary_real=NUM(Data_line$[1,1])
24160         ELSE
24165               Temporary_real=NUM(Data_line$[2,2]) ! Skip the "-"
24170         END IF    ! of not "-"
24175         !
24180         ! Check if the corresponding ASCII of the first char
24185         ! is > then 9 or < then 0
24190         !
24195         IF (Temporary_real>NUM("9")) OR (Temporary_real<NUM("0")) THEN
24200               ! This is a character not a number
24205               Ok_input$="FALSE"
24210         ELSE     ! This is a numeric string
24215               ENTER Data_line$;Temporary_real
24220               Ok_input$="TRUE "
24225         END IF     ! of numeric
24230    END IF ! not ien
24235    RETURN Ok_input$
24240 !
24245 FNEND ! of Is_string_num
24250 ! -------------------------End-of-Sub-Function-------------------------
24255 !
24260 ! ***************************************************************************
24265 DEF FNDefine_used_mra(Requested_mra) !  MRA AS DEFINED IN LOOK-UP TABLE *
24270 ! ***************************************************************************
24275 !                                                                        *
24280 ! SUBFUNCTION NAME: DEFINE USED MRA              DATE: 29 SEPTEMBER 1985 *
24285 !                                                                        *
24290 ! PROGRAMMER : SYLVAIN FLEURANT                 DATE: 29 SEPTEMBER 1985 *
24295 !                                                                        *
```

```
24300 ! PURPOSE : To return the value of a MRA which is defined in the        *
24305 !           MRA Look-up table using a Requested value of MRA.           *
24310 !                                                                       *
24315 ! INPUT: Requested_mra                                                  *
24320 ! OUTPUT: Used_mra                                                      *
24325 ! SIDE EFFECTS: NIL                                                    *
24330 ! REMARKS: A REAL VALUE IS RETURNED IN DEGREES.                         *
24335 ! ***************************************************************************
24340   !
24345   INTEGER Mra_value              ! Result of the function
24350   !
24355   IF ABS(Requested_mra)>=170 THEN
24360      IF Requested_mra<0 THEN
24365         Mra_value=-170
24370      ELSE
24375         Mra_value=170       ! Table Look-up define for up to +/- 170.
24380      END IF
24385   ELSE
24390      IF ABS(Requested_mra)>=165 THEN
24395         Mra_value=INT(Requested_mra)    ! Defined for every degrees
24400      ELSE
24405         IF ((ABS(Requested_mra MOD 5))<2.5) THEN
24410            Mra_value=5*(Requested_mra DIV 5) ! Define every 5 degrees
24415         ELSE
24420            IF Requested_mra>=0 THEN
24425               Mra_value=5*((Requested_mra+5) DIV 5)
24430            ELSE
24435               Mra_value=5*((Requested_mra-5) DIV 5)
24440            END IF ! >= 0
24445         END IF  ! MOD 5 < 2.5
24450      END IF ! >= 165
24455   END IF ! ABS( Requested_mra) >= 170
24460   RETURN Mra_value
24465   !
24470 FNEND ! Define_used_mra
24475 !
24480 ! -------------------------End-of-Sub-Function-------------------------
24485 !
24490 ! ***************************************************************************
24495 SUB Comp_axis_pplot(Psi(*),Xaxis_db_bp(*),Yaxis_db_bp(*),Mag_db_bp(*),Mini
mum_nn_db)    ! COMPUTE THE VALUE OF X AND Y USED FOR THE POLAR PLOT          *
24500 ! ***************************************************************************
24505 !                                                                       *
24510 ! SUBPROGRAM NAME: COMPUTE AXIS POLAR PLOT       DATE: 29 SEPTEMBER 1985 *
24515 !                                                                       *
24520 ! PROGRAMMER: SYLVAIN FLEURANT                  DATE: 29 SEPTEMBER 1985 *
24525 !                                                                       *
24530 ! PURPOSE: To compute the values of the X-axis and Y-axis to be used to  *
24535 !          plot a polar plot in dB for angle between -180 to 180.       *
24540 !                                                                       *
24545 ! INPUT : Psi, Xaxis_db_bp, Yaxis_db_bp, Mag_db_bp, Minimum_nn_db       *
24550 ! OUTPUT: Xaxis_db_bp, Yaxis_db_bp.                                     *
24555 ! SIDE EFFECTS: NIL                                                    *
24560 ! REMARKS: The angles must vary between -180 to 180 degrees; otherwise, *
24565 !          computationnal errors will occur. Psi must be in degrees.    *
24570 ! ***************************************************************************
24575   !
24580     DEG
24585     FOR I=-180 TO 180 STEP 1
24590       IF (Mag_db_bp(I)<=Minimum_nn_db) THEN
```

```
24595          Xaxis_db_bp(I)=0
24600          Yaxis_db_bp(I)=0
24605       ELSE
24610        IF (Psi(I)<=-90) THEN
24615             Xaxis_db_bp(I)=(Minimum_nn_db-Mag_db_bp(I))*SIN(180+Psi(I))
24620             Yaxis_db_bp(I)=(Minimum_nn_db-Mag_db_bp(I))*COS(180+Psi(I))
24625        ELSE
24630            IF (Psi(I)<=0) THEN
24635               Xaxis_db_bp(I)=(Minimum_nn_db-Mag_db_bp(I))*COS(90+Psi(I))
24640               Yaxis_db_bp(I)=(-Minimum_nn_db+Mag_db_bp(I))*SIN(90+Psi(I))
24645            ELSE
24650               IF (Psi(I)<=90) THEN
24655                  Xaxis_db_bp(I)=(-Minimum_nn_db+Mag_db_bp(I))*COS(90-Psi(I)
)
24660                  Yaxis_db_bp(I)=(-Minimum_nn_db+Mag_db_bp(I))*SIN(90-Psi(I)
)
24665               ELSE   ! FOR UP TO MAX_ANGLE i.e. PI
24670                  Xaxis_db_bp(I)=(-Minimum_nn_db+Mag_db_bp(I))*COS(Psi(I)-90
)
24675                  Yaxis_db_bp(I)=(Minimum_nn_db-Mag_db_bp(I))*SIN(Psi(I)-90)
24680               END IF  !( PSI <= 90 )
24685            END IF  ! (PSI <= 0)
24690         END IF ! (PSI <=-90)
24695       END IF ! Magnitude is <= Minimum_nn_db
24700      NEXT I
24705 !
24710    SUBEND ! Comp_axis_pplot
24715 ! -------------------------End-of-Sub-Program--------------------------
24720 !
24725 ! ****************************************************************************
24730 SUB Comp_mag_in_db(Mag_db_bp(*),Magnitude_bp(*),Minimum_nn_db) !          *
24735 !       COMPUTE THE NORMALIZE MAGNITUDE OF BEAM PATTERN IN dB            *
24740 ! ****************************************************************************
24745 !                                                                          *
24750 ! SUBPROGRAM NAME: COMPUTE MAGNITUDE IN dB        DATE: 29 SEPTEMBER 1985 *
24755 !                                                                          *
24760 ! PROGRAMMER: SYLVAIN FLEURANT                   DATE: 29 SEPTEMBER 1985 *
24765 !                                                                          *
24770 ! PURPOSE: To compute the value of the normalize beam pattern in decibel *
24775 !          from 0 dB and down up to minimum_nn_db.                         *
24780 !                                                                          *
24785 ! INPUT : Mag_db_bp, Magnitude_bp, Minimum_nn_db.                         *
24790 ! OUTPUT: Mag_db_bp                                                       *
24795 ! SIDE EFFECTS: NIL                                                      *
24800 ! REMARKS: 1. The arrays are indexed from -180 to 180.                   *
24805 !             This value can also be implemented using Max_value as an   *
24810 !              additional parameter.                                       *
24815 ! ****************************************************************************
24820  !
24825        Minimum_nn_mag=10^(Minimum_nn_db/10)
24830        FOR I=-180 TO 180 STEP 1
24835             IF ABS(Magnitude_bp(I))<=(Minimum_nn_mag) THEN
24840                  Mag_db_bp(I)=Minimum_nn_db
24845             ELSE
24850                  Mag_db_bp(I)=20*LOG(ABS(Magnitude_bp(I)))/LOG(10)
24855             END IF
24860        NEXT I
24865  !
24870  SUBEND ! Comp_mag_in_db
24875 ! -------------------------End-of-Sub-Program--------------------------
```

```
24880 !
24885 ! ************************************************************************
24890 DEF FNUser_sta_m_choi$ ! ENTER USER STAVE MENU CHOICE FOR EXECUTION      *
24895 ! ************************************************************************
24900 !                                                                        *
24905 ! FUNCTION NAME: USER_STAVE_MENU_CHOICE          DATE:  5 NOVEMBER 1985 *
24910 !                                                                        *
24915 ! PROGRAMMER: Sylvain Fleurant                  DATE:  5 NOVEMBER 1985 *
24920 !                                                                        *
24925 ! PURPOSE : To display a message requesting the user weither he wants   *
24930 !           to use the ind_stave menu or not, i.e. does he wants to     *
24935 !           change any individual complex sensitivities of any hydro-   *
24940 !           phones.                                                      *
24945 !                                                                        *
24950 ! INPUT: NIL                                                           *
24955 ! OUTPUT: TRUE or FALSE return to Stave_menu_flag$                      *
24960 ! SIDE EFFECTS: Change the key functions. Also, clear the screen.       *
24965 ! REMARKS: NIL                                                         *
24970 ! ************************************************************************
24975 !
24980 !  LOCAL DATA BASE
24985    ALLOCATE Clear$[2]           ! Used to clear the screen
24990    ALLOCATE User_choice$[5]     ! Answer of user to use the menu or not
24995                                 ! TRUE = use the menu
25000                                 ! FALSE = proceed without using the menu
25005    !
25010    Clear$=CHR$(255)&CHR$(75)    ! = (SHIFT)(CLR SCR) key on the keyboard
25015    User_menu$="FALSE"
25020    !
25025   PRINT USING "//"
25030   PRINT "IT IS POSSIBLE TO CHANGE MANUALLY"
25035   PRINT "THE INDIVIDUAL COMPLEX SENSITIVITIES (MAGNITUDE AND PHASE)"
25040   PRINT "OF ANY HYDROPHONES USED AT A GIVEN MRA"
25045   PRINT USING "//"
25050   PRINT "PRESS K0 TO USE THIS MENU"
25055   PRINT "PRESS K1 TO PROCEED WITHOUT ANY CHANGE"
25060   ! ---------------------------------------------------------------------
25065   !
25070   ! DEFINE KEY CHOICE
25075   !
25080   CONTROL 1,12;0       ! Activate the softkey functions and display them
25085   ON KEY 0 LABEL "CHANGE RQD" GOTO Change_rqd_lbl
25090   ON KEY 1 LABEL "NO CHANGE" GOTO No_change_lbl
25095   ON KEY 2 GOTO Not_used_st_lbl
25100   ON KEY 3 GOTO Not_used_st_lbl
25105   ON KEY 4 GOTO Not_used_st_lbl
25110   ON KEY 5 GOTO Not_used_st_lbl
25115   ON KEY 6 GOTO Not_used_st_lbl
25120   ON KEY 7 GOTO Not_used_st_lbl
25125   ON KEY 8 GOTO Not_used_st_lbl
25130   ON KEY 9 GOTO Not_used_st_lbl
25135   !
25140   ! ---------------------------------------------------------------------
25145   !
25150 Spin_staves1:  GOTO Spin_staves1     ! WAIT for a softkey interrupt,i.e.,
25155                                      ! Operator I/O
25160   !
25165   ! ---------------------------------------------------------------------
25170   !
25175 Change_rqd_lbl: !
```

```
25180       User_choice$="TRUE "
25185       GOTO End_sta_m_choi
25190     !
25195     ! ------------------------------------------------------------------
25200     !
25205 No_change_lbl: !
25210       User_choice$="FALSE"
25215       GOTO End_sta_m_choi
25220     !
25225     ! ------------------------------------------------------------------
25230     !
25235 Not_used_st_lbl:  ! NOT USED - STAVE MENU KEY
25240       BEEP 500,.5
25245       GOTO Spin_staves!
25250     ! ------------------------------------------------------------------
25255     !
25260     PAUSE
25265     !
25270 End_sta_m_choi:  !
25275     CONTROL 1,12;1              ! Turn off the softkeys and its display.
25280     OUTPUT 2;Clear$;            ! Clear the screen before leaving
25285     RETURN User_choice$
25290     !
25295   FNEND ! of User_sta_m_choi$
25300   ! -------------------------End-of-Function ----------------------------
25305   !
25310   ! ************************************************************************
25315 SUB Angle_defn(Used_mra,From_angle_1,To_angle_1,From_angle_2,To_angle_2)!
25320   ! ************************************************************************
25325   !                                                                       *
25330   ! SUBPROGRAM NAME: ANGLE_DEFN                      DATE: 26 NOVEMBER 1985 *
25335   !                                                                       *
25340   ! PROGRAMMER: Sylvain Fleurant                     DATE: 26 NOVEMBER 1985 *
25345   !                                                                       *
25350   ! PURPOSE: To define the angles to be used for the visible region of    *
25355   !          the polar plot.                                              *
25360   !                                                                       *
25365   ! INPUT: Used_mra, From_angle_1, To_angle_1, From_angle_2, To_angle_2   *
25370   ! OUTPUT: From_angle_1, To_angle_1, From_angle_2, To_angle_2            *
25375   ! SIDE EFFECTS: NIL                                                    *
25380   ! REMARKS: It assumes that the maximum angles are +/- 180 Degrees.     *
25385   !          Return real values are in degrees.                           *
25390   !          Visible region is always Used_mra - 90 degrees to Used_Mra + *
25395   !          90 degrees.                                                  *
25400   !          The FROM angle is always smaller than the TO angle.          *
25405   ! ************************************************************************
25410   !
25415     IF (Used_mra>=-90) AND (Used_mra<=90) THEN
25420           From_angle_1=INT(Used_mra-90)
25425           To_angle_1=INT(Used_mra+90)
25430           From_angle_2=0
25435           To_angle_2=0   ! 1 will cover everything
25440     ELSE
25445         IF (Used_mra<-90) THEN
25450            From_angle_1=-180
25455            To_angle_1=INT(Used_mra+90)
25460            From_angle_2=270+INT(Used_mra)
25465            To_angle_2=180
25470         ELSE  ! (USED_MRA > 90)
25475            From_angle_1=INT(Used_mra-90)
```

```
25480           To_angle_1=180
25485           From_angle_2=-180
25490           To_angle_2=-270+INT(Used_mra)
25495         END IF ! (used_mra < - 90)
25500       END IF  ! (Used_mra > -90 AND < 90)
25505       !
25510       !PRINT "USED MRA      = ";Used_mra
25515       !PRINT "FROM ANGLE 1 = ";From_angle_1
25520       !PRINT "TO ANGLE 1   = ";To_angle_1
25525       !PRINT "FROM ANGLE 2 = ";From_angle_2
25530       !PRINT "TO ANGLE 2   = ";To_angle_2
25535       !
25540       SUBEND ! Angle_defn
25545     ! ----------------------END-OF-SUBPROGRAM-----------------------------
25550     !
25555     ! ***************************************************************************
25560     SUB Read_actual_cs(Freq,Amp_wg_buffer(*),Pha_wg_buffer(*),Minimum_nn_db,M
ax_nn_staves) !
25565     ! ***************************************************************************
25570     ! SUB-PROGRAM: READ ACTUAL COMPLEX SENSITIVITY   DATE : 8 DECEMBER 1985 *
25575     !                                                                        *
25580     ! PROGRAMMER: Sylvain Fleurant                                           *
25585     !                                                                        *
25590     ! PURPOSE: To read the actual complex sensitivities of all staves for a *
25595     !          given frequency. The magnitude is converted from decibels to *
25600     !          linear scale, and the phase from degrees to radians.          *
25605     !                                                                        *
25610     ! INPUT: Frequency, Amp_wg_buffer, Pha_wg_buffer, Minimum_nn_db,         *
25615     !        Max_nn_staves.                                                  *
25620     ! OUTPUT: Amp_wg_buffer, Pha_wg_buffer                                   *
25625     ! SIDE EFFECTS: NIL                                                      *
25630     ! REMARKS: 1. Files are stored as a letter F followed by a frequency    *
25635     !             value: e.g., F1100.                                        *
25640     !          2. The values of frequency vary between 50 to 5000 Hz:        *
25645     !             a.  50 to 100, by steps of 25;                             *
25650     !             b.  150 to 1000, by steps of 50;                           *
25655     !             c.  1100 to 2500, by steps of 100; and                     *
25660     !             d.  2750 to 5000, by steps of 250.                         *
25665     !                                                                        *
25670     ! ***************************************************************************
25675     ! LOCAL DATA BASE
25680     !
25685       INTEGER Frequency       ! final value to be used for the disk file
25690       INTEGER Freq_offset     ! Offset to add to Freq
25695       INTEGER I,J             ! For indexing purposes
25700       !
25705       ! Freq_file$ is the name of the file to be read
25710       ! Label$ id the name of the label of the disc of complex sensitivities
25715       !
25720       !INPUT "ENTER THE FREQUENCY ?",Freq
25725       !PRINT "THE ENTERED FREQUENCY IS : ",Freq
25730       !
25735       ! ---------------------------------------------------------------------
25740       !
25745       ! DEFINE THE FREQUENCY TO USE TO ACCESS THE FILE
25750       !
25755     IF Freq<=62.5 THEN
25760        Frequency=50
25765     ELSE
25770      IF Freq<125 THEN
```

```
25775      ! Every 25 Hz
25780      IF Freq<87.5 THEN
25785          Frequency=75
25790      ELSE
25795          ! It is 100 hZ
25800          Frequency=100
25805      END IF ! < 87.5
25810      ! ---------------------------------------------------------------------
25815    ELSE
25820      IF Freq<1050 THEN
25825         !  EVERY 50 Hz
25830         !
25835         IF Freq<975 THEN
25840             Frequency=10*PROUND(Freq/10,1)
25845             IF ((Frequency-Freq)<0) THEN
25850               IF ((Frequency-Freq)<=-25) THEN
25855                  Freq_offset=50
25860               ELSE
25865                  Freq_offset=0
25870               END IF  ! (Frequency-Freq) <= -25
25875             ELSE
25880               IF ((Frequency-Freq)<25) THEN
25885                  IF ((Frequency-Freq)>0) THEN
25890                     Freq_offset=0
25895                  END IF
25900               ELSE
25905                   Freq_offset=-50
25910               END IF  ! (Frequency-Freq) < 25
25915             END IF ! ((Frequency-Freq)<0)
25920           Frequency=10*PROUND(Freq/10,1)+Freq_offset
25925           Freq_offset=0
25930         ELSE
25935            Frequency=1000
25940         END IF ! < 975
25945      ! ---------------------------------------------------------------------
25950       ELSE
25955         IF Freq<2625 THEN
25960          ! EVERY 100 Hz
25965           IF Freq<=2450 THEN
25970           !
25975              IF ((Freq MOD 100)<50) THEN
25980                  Frequency=100*INT(Freq DIV 100)
25985              ELSE
25990                  Frequency=100+100*INT(Freq DIV 100)
25995              END IF
26000           ELSE
26005              Frequency=2500
26010           END IF ! < 2450
26015         ! ---------------------------------------------------------------------
26020         ELSE
26025           IF Freq<4875 THEN
26030            ! EVERY 250 Hz
26035            IF Freq<2875 THEN
26040                Frequency=2750
26045            ELSE
26050                Frequency=100*PROUND(Freq/100,1)
26055                IF ((Frequency-Freq)<0) THEN
26060                  IF ((Frequency-Freq)<-375) THEN
26065                      Freq_offset=500
26070                  ELSE
```

```
26075                   IF ((Frequency-Freq)<-126) THEN
26080                      Freq_offset=250
26085                   ELSE
26090                      Freq_offset=0
26095                   END IF ! < -126
26100                END IF ! < -375
26105              ELSE
26110                IF ((Frequency-Freq)>375) THEN
26115                     Freq_offset=-500
26120                ELSE
26125                    IF ((Frequency-Freq)>=250) THEN
26130                        Freq_offset=-250
26135                    ELSE
26140                        Freq_offset=0
26145                    END IF  ! >= 250
26150                END IF ! < 375
26155              !
26160              END IF ! (Frequency-Freq)<0
26165              Frequency=100*PROUND(Freq/100,1)+Freq_offset
26170            END IF ! of freq < 2875
26175          ! ------------------------------------------------------------
26180          ELSE
26185           ! FOR 4875 AND UP
26190           Frequency=5000
26195          END IF ! < 4875
26200        END IF ! <2625
26205      END IF ! <1050
26210    END IF ! <125
26215   END IF ! <= 62.5
26220    !
26225    ! PRINT "THE FREQUENCY TO BE USED IS : ",Frequency
26230    !
26235    ! ------------------------------------------------------------------
26240    !
26245    MASS STORAGE IS ":INTERNAL,4,0"       !RIGHT SIDE
26250    !
26255 Insert_disc:!
26260    PRINT USING "//"
26265    PRINT "INSERT THE FLOPPY DISK"
26270    PRINT "OF RELATIVE COMPLEX SENSITIVITY PER FREQUENCY"
26275    PRINT "IN THE RIGHT DISK DRIVER"
26280    PRINT " "
26285    PRINT "Press CONTINUE when it is done"
26290    PAUSE
26295    PRINT USING "@"
26300    READ LABEL Label$ FROM ":INTERNAL"
26305    IF Label$<>"FREQ" THEN
26310        DISP "You have inserted an incorrect disc"
26315        BEEP
26320        WAIT 3
26325        GOTO Insert_disc
26330    END IF
26335    DISP "PROCEEDING"
26340  !
26345  ! ------------------------------------------------------------------
26350  !
26355       Freq_file$="F"&VAL$(Frequency)
26360       DISP "READING FILE :",Freq_file$
26365     ! ON ERROR GOSUB New_disc
26370       ASSIGN @Path! TO Freq_file$   ! Open the file
```

```
26375       FOR I=1 TO Max_nn_staves STEP 1
26380         ENTER @Path1;Amp_wg_buffer(I),Pha_wg_buffer(I)
26385       NEXT I
26390       ASSIGN @Path1 TO *    ! Close it
26395       DISP " "
26400       GOTO Term_cw_io
26405  !
26410  ! ************************************************************************
26415 New_disc:  ! A new disc is needed
26420  ! ************************************************************************
26425   !DISP "Enter a new Floppy disc, then press CONTINUE"
26430       PURGE Freq_file$
26435       BEEP 999,.1
26440  RETURN ! New_disc
26445  !
26450  ! ------------------------------------------------------------------------
26455 Term_cw_io: !
26460  ! ------------------------------------------------------------------------
26465    ! CONVERT IN LINEAR SCALE AND RADIANS
26470    MAT Pha_wg_buffer= Pha_wg_buffer*(3.141592654/180)
26475    FOR I=1 TO Max_nn_staves
26480       IF Amp_wg_buffer(I)<Minimum_nn_db THEN
26485           Amp_wg_buffer(I)=0.
26490       ELSE
26495           Amp_wg_buffer(I)=10^(Amp_wg_buffer(I)/20)  ! from dB to linear
26500       END IF
26505    NEXT I
26510    MASS STORAGE IS ":INTERNAL,4,1"      !LEFT SIDE again
26515  SUBEND ! of Read_actual_cs
26520  ! -------------------------END-OF-SUBPROGRAM--------------------------
```

LIST OF REFERENCES

1. Wilson, O. B., An Introduction to the Theory and Design of Sonar Transducers, US Naval Postgraduate School, Monterey, CA, To be published in 1986.

2. Ziomek, L. J. , Underwater Acoustics, A Linear Systems Theory Approach, Academic Press, Inc., Orlando, FLA, 1985, 290 pp.

3. Steinberg, Bernard D. , Principles of Aperture and Array System Design including Random and Adaptive Arrays, John Wiley & Sons, 1976, 356 pp.

4. Kinsler, L.E., Frey, A.R., Coppens, A.B., and Sanders, J.V., Fundamentals of Acoustics, 3rd edition, John Wiley & Sons, 1980, 479 pp.

5. Oppenheim, Alan V., and Schafer, R.V., Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975, 585 pp.

6. Brigham, E.O. , The Fast Fourier Transform , Prentice-Hall, Inc., Englewood Cliffs, N.J., 1979, 252pp.

7. Self, R.E., Automated Test and Evaluation of Passive Submarine Sonar Transducers M.S. Thesis, US Naval Postgraduate School, Monterey, California, Dec 1985.

8. Muller, Black, and Dunn, Journal of Acoustical Society of America, Vol 10, No 1, p. 6, 1938.

9. Olson, F. Harry, Elements of Acoustical Engineering, D. Van Nostrand Company, Inc, 1947, 539 pp.

10. Boehm, W. Barry, Software Engineering Economy, Prenticew-Hall Inc, Englewoods Cliffs, N.J., 1981, 767 pp.

11. Fairley, E. Richard Software Engineering Concepts, McGraw-Hill Book Company, 1985, 364 pp.

12. Urick, J. Robert, Principles of Underwater Acoustics, 3rd Edition, McGraw-Hill Book Company, 1983, 423 pp.

13. MacLennan, J. Bruce, Principles of Programming Languages, Holt, Rinhart, and Winston, 1983, 544 pp.

14. Horowitz, Ellis, Fundamentals of Programming Languages Computer Science Press, 1983, 450 pp.

15. Myers, J. Glenford, Reliable Software through Composite Design, Van Nostrand Reinhold Company, 1975, 159 pp.

16. Constantine, L. and Yourdon, E., Structured Design: Fundamentals of A Discipline of Computer Program and Systems Design, Prentice-Hall, Englewood Cliffs, N.J., 1979.

17. Schneider, G. M. and Bruel, S.C., Advanced Programming and Problem Solving with Pascal, John Wiley & Sons, 1981, 506pp.

BIBLIOGRAPHY

Albers R. J., Underwater Acoustics, Plenum Press, 1963.

Albers R. J., Underwater Acoustics Volume 2, Plenum Press, 1967.

Boehm Barry W., "Seven Basic Principles of Software Engineering", Journal of Systems and Software, Volume 3, Number 1, 1983, pp. 3-24.

Bobber R.J., Underwater Electroacoustic Measurements, Naval Research Laboratory, July 1970.

Camp L., Underwater Acoustics, Wiley, 1970.

Dahl, Ole-Johan, Dijkstra, Edger W., and Hoare, C.A.R., Structured Programming, Academic Press, NY, 1972.

Handbook of Array Design Technology Volume 1, by Applied Hydro-Acoustics Research, Inc, for Naval Electronics Systems Command, June 1976.

Hewlett-Packard Company, BASIC 3.0 Graphics Techniques for the HP9000 Series 200 Computers, 1984.

Hewlett-Packard Company, BASIC 3.0 Language Reference for the HP9000 Series 200 Computers, 1984.

Hewlett-Packard Company, BASIC 3.0 Programming Techniques for the HP9000 Series 200 Computers, 1984.

Hunt E.V., Electroacoustics, The Analysis of Transduction, and Its historical Background, American Institute of Physics for the Acoustical Society of America, 1976

IEEE Standard Glossary of Software Engineering Terminology, IEEE Standard 729-1983, The Institute of Electrical and Electronics Engineers, Inc., New-York, 1983.

Leblanc C.L., Handbook of hydrophone Element Design Technology, Naval Undersea Systems Center Technical Document 5813, Oct 1978.

Miller E., and Howden W.E., Tutorial: Software Testing & Validation Techniques, IEEE Computer Society, Long Beach, CA, 1978.

Morse P.M., Vibration and Sound, American Institute of Physics for the Acoustical Society of America, 1982

Peters L. J., Software Design: Methods and Techniques, Yourdon Press, NY, 1981.

Parnas D.L., "On the Criteria to be Used in Decomposing Systems into Modules", Communications of the ACM, Volume 15, Number 12, December 1972, pp. 1053 - 1058.

Parnas D.L., "Designing Software for Ease of Extension and Contractions.", IEEE Transactions on Software Engineering, Volume SE-5, Number 2, March 1979, pp. 128 - 138.

Putnam L. H., IEEE Tutorial: Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers, IEEE Computer Society Press, Los Alamitos, CA, 1980, 100 pp.

INITIAL DISTRIBUTION LIST

No.		Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library Code 0142 Naval Postgraduate School Monterey, CA 93943-5100	2
3.	Commanding Officer ATTN: LCDR J. Markevicz Naval Underwater Systems Command Newport, Rhode Island 02841	2
4.	Commander, Naval Sea Systems Command ATTN: C. Clark SEA 63X5 Washington, D.C. 20362	1
5.	Commander, Naval Sea Systems Command ATTN: O5NE58 (C. Allen) Washington, D.C. 20362	2
6.	Commanding Officer Naval Research Laboratory Underwater Sound Reference Detachment ATTN: Dr. R. W. Timme (code 5870) P. O. Box 8337	1
7.	Officer in Charge (LCDR J. Johnson) Naval Underwater Systems Center AUTEC West Palm Beach Detachment West Palm Beach, Florida 33403	1
8.	Professor O. B. Wilson (Code 61Wl) Naval Postgraduate School Monterey, California 93943	10
9.	Professor S. L. Garrett (Code 61Gx) Naval Postgraduate School Monterey, California 93943	6
10.	Professor L. J. Ziomek (Code 62Zm) Naval Postgraduate School Monterey, California 93943	6
11.	Professor R. W. Hamming (Code 52Hg) Naval Postgraduate School Monterey, California 93943	1
12.	Chairman, Department of Physics (Code 61) Naval Postgraduate School Monterey, California 93943	1
13.	Chairman, Department of Computer Science (Code 52) Naval Postgraduate School Monterey, California 93943	1

14. Texas Research Institute, Inc. 1
ATTN: L. Smith
9063 Bee Caves Road
Austin, Texas 78733-6201

15. Applied Research Laboratories 1
The University of Texas at Austin
ATTN: D. D. Baker
P. O. Box 8029
Austin, Texas 78733-6201

16. Commander, Submarine Squadron Eighteen 1
SMMS/PMT Detachment
ATTN: Chief Seal, Sonar Division
Charleston, South Caroline 29408

17. Capt S. J. Fleurant 6
1420 Via Marettimo
Monterey, California 93940

18. LT R. E. Self 1
112 Lamboum Wayne
Greenwills, South Carolina 29615

19. Commanding Officer 1
ATTN: Mr. W. Anderson (c/o Code 701)
Naval Undersea Warfare Engineering Station
Keyport, Washington 98345

20. LCDR C. Burmaster 1
Physics Department, (Code 9C)
United States Naval Academy
Anapolis, Maryland 21402

21. Commanding Officer 1
Naval Research Laboratory
Underwater Sound Reference Detachment
ATTN: C. Ruggiero (Code 5970)
P. O. Box 8337
Orlando, Florida 32856-8337

22. Commanding Officer 1
Naval Weapons Support Center
ATTN: Thomas D. Peter, Bldg 2530
Crane, Indiana 47522

23. Commanding Officer 1
Naval Weapons Support Center
ATTN: Ann Wilson, Bldg 2530
Crane, Indiana 47522

24. Commanding Officer 1
Naval Underwater Systems Center
ATTN: Kenneth Webman (Code 3234)
New London, Connecticut 06320

25. Commander, Submarine Squadron Sixteen 1
SMMS/PMT Detachment
Naval Submarine Base
King's Bay, Georgia 31547

26. LCDR D. V. Conte 1
NUSC-Fort Trumbull
27 Wabash St. #2
New London, Connecticut 06320

27. LCDR L. J. Skowronek 1
NAVFAC Centerville Beach
Ferndale, California 95536

28. Commander, Naval Sea Systems Command — 1
ATTN: LCDR J. Mason
SEA.PMS. 409-A3
Washington, D.C. 20362

29. Professor S.R. Baker (Code 61Xi) — 1
Naval Postgraduate School
Monterey, California 93943

30. Commanding Officer — 2
Aurora Software Development Unit
Canadian Forces Base Greenwood
Greenwood, Nova-Scotia, Canada B0P-1N0

31. LCDR M. Tunnicliffe — 1
c/o Commandant
ATTN: Combat Division
Canadian Forces Fleet School Halifax
F.M.O. Halifax
Nova-Scotia, Canada B3K-2X0

32. Canadian Forces Technical Library — 1
National Defence Headquarters
Ottawa, Ontario, Canada K1A-0K2

33. DGMAEM/DMAEM 4-2 — 2
ATTN: Maj B. Vermander
National Defence Headquarters
Ottawa, Ontario, Canada K1A-0K2

34. DPED — 1
National Defence Headquarters
Ottawa, Ontario, Canada K1A-0K2

35. Commandant — 1
ATTN: H.J. Duffus
Dean, Science and Engineering
Royal Roads Military College
FMO Victoria, British Columbia, Canada V0S-1B0

36. Mr Jacques Fleurant — 1
10680 rue Hamel
Ahuntsic, Quebec, Canada H2C-2X5

37. Mr Richard Boudreault — 1
Canadian Astronautics Limited
Space Division
1050 Morrison Drive
Ottawa, Ontario, Canada K2H-8K7